IBM Distributed Computing Environment for AIX,
Version 2.2:

Distributed File Service Administration Guide and Reference

IBM Distributed Computing Environment for AIX,
Version 2.2:

Distributed File Service Administration Guide and Reference

Note

Before using this document, read the general information under “Appendix. Notices” on page 763.

First Edition (February 1998)

This edition applies to Version 2.2 of IBM Distributed Computing Environment for AIX and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. Send your comments to the following address:

International Business Machines Corporation

Department VLXA

11400 Burnet Road

Austin, Texas

78758

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Preface

The *IBM DCE for AIX, Version 2.2: DFS Administration Guide and Reference* serves two purposes:

- It provides concepts and procedures that enable you to manage the Distributed File Service (DFS) in your Distributed Computing Environment (DCE) cell.
- It provides detailed reference information to help you learn more about the complete syntax and use of each DFS command and configuration file.

Audience

This guide and reference is written for system and network administrators who have previously administered a UNIX environment.

Applicability

This revision applies to the IBM® DCE for AIX, Version 2.2 (DCE 2.2 for AIX). See your software license for details.

Purpose

The purpose of this guide and reference is to help system and network administrators plan, configure, and manage DFS in a DCE cell. After you have initially installed and configured DCE and DFS in your cell, refer to this document for information about expanding and maintaining your DFS configuration. Also refer to this document for complete descriptions of all DFS commands.*IBM DCE for AIX, Version 2.2: Quick Beginnings* contain instructions for installing and building DCE source code, and it contains release-specific information about DFS.

Document Usage

The *IBM DCE for AIX, Version 2.2: DFS Administration Guide and Reference* is divided into the following parts:

- Part 1. IBM DCE DFS Administration Guide
  - Chapter 1. An Overview of DFS
  - Chapter 2. DFS Configuration Issues
  - Chapter 3. Using ACLs and Groups
  - Chapter 4. Using Administrative Lists and Keytab Files
  - Chapter 5. Monitoring and Controlling Server Processes
  - Chapter 6. Making Filesets and Aggregates Available
  - Chapter 7. Managing Filesets
  - Chapter 8. Configuring the Cache Manager
  - Chapter 9. Configuring the Backup System
  - Chapter 10. Backing Up and Restoring Data
  - Chapter 11. Monitoring and Tracing Tools
- Part 2. IBM DCE DFS Administration Reference
  - Chapter 12. Configuration Files
  - Chapter 13. Configuration Commands

– Chapter 14. Administrative Commands

Related Documents

For additional information about IBM DCE 2.2, refer to the following documents:

- *IBM DCE for AIX, Version 2.2: Introduction to DCE*
- *IBM DCE for AIX, Version 2.2: Administration Guide—Introduction*
- *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components*
- *IBM DCE for AIX, Version 2.2: Administration Commands Reference*
- *IBM DCE for AIX, Version 2.2: Application Development Guide—Introduction and Style Guide*
- *IBM DCE for AIX, Version 2.2: Application Development Guide—Core Components*
- *IBM DCE for AIX, Version 2.2: Application Development Guide—Directory Services*
- *IBM DCE for AIX, Version 2.2: Application Development Reference*
- *IBM DCE for AIX, Version 2.2: Problem Determination Guide*
- *IBM DCE for AIX, Version 2.2: Release Notes*
- *IBM DCE for AIX, Version 2.2: Quick Beginnings*
- *IBM DCE for AIX, Version 2.2: High Availability Cluster Multi-Processing Guide for DCE and DFS*

Typographic and Keying Conventions

This guide uses the following typographic conventions:

Bold **Bold** words or characters represent system elements that you must use literally, such as commands, options, and pathnames.

Italic *Italic* words or characters represent variable values that you must supply. *Italic* type is also used to introduce a new DCE term.

Constant width
Examples and information that the system displays appear in `constant width` typeface.

[ ] Brackets enclose optional items in format and syntax descriptions.

{ } Braces enclose a list from which you must choose an item in format and syntax descriptions.

| A vertical bar separates items in a list of choices.

< > Angle brackets enclose the name of a key on the keyboard.

... Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

dcelocal
The OSF variable *dcelocal* in this document equates to the AIX variable */opt/dcelocal*.

dceshared
The OSF variable *dceshared* in this document equates to the AIX variable */opt/dcelocal*.

This guide uses the following keying conventions:

<Ctrl- *x***>** **or** ^ *x*

The notation **<Ctrl-** *x***>** or ^ *x* followed by the name of a key indicates a control character sequence. For example, **<Ctrl-C>** means that you hold down the control key while pressing **<C>**.

<Return>

The notation **<Return>** refers to the key on your terminal or workstation that is labeled with the word Return or Enter, or with a left arrow.

Pathnames of Directories and Files in DCE Documentation

For a list of the pathnames for directories and files referred to in this guide, see the *IBM DCE for AIX, Version 2.2: Administration Guide—Introduction*.

Part 1. IBM DCE DFS Administration Guide

Chapter 1. An Overview of DFS

This chapter introduces basic concepts of the DCE Distributed File Service (DFS). It provides introductory information about the components of DFS, the administrative advantages they offer, and their interaction with other DCE components. It also provides a brief overview of some common DFS administrative tasks, explains the DFS command structure, and describes how you can get help for DFS commands. You should read and understand this chapter before performing any of the tasks detailed in the Guide part of this guide and reference.

Features of DFS

DCE DFS is a distributed client/server application that presents DCE with a global view of a set of files and directories (a file system), independent of machine boundaries. This global view is called the *DFS filespace*.

DFS is considered distributed because files can be physically stored on many different machines, but they are available to users on every machine. DFS allows users to share files stored on computers in a network as easily as files stored on a local machine. Despite this distribution of files, it still appears to users that there is a single filespace.

DFS Server Machines

DFS server machines run processes that provide services such as making data available and monitoring and controlling other processes. They are categorized by the processes they run (that is, the roles they assume). For example, a server machine that runs the processes necessary for storing and exporting data assumes the role of a File Server machine. The processes of a File Server machine include the Fileset Server, which provides an interface to the DFS commands and components used to manipulate filesets, and the File Exporter, which runs in a modified kernel to make DFS files available to the global namespace.

Other server machine roles include a System Control machine that updates other server machines with identical versions of system configuration files; Binary Distribution machines that distribute system binaries to other machines with the same CPU/operating system type; Fileset Database machines that house the master and replica versions of the Fileset Location Database (FLDB) where information about the location of system and user files is maintained; and Backup Database machines that house the master and replica versions of the Backup Database where information used to back up and restore system and user files resides. (See “Chapter 2. DFS Configuration Issues” on page 27 for more information about the roles of DFS server machines.)

DFS Client Machines

DFS client machines provide computational power, access to DFS files, and other general-purpose tools. In some configurations, a server machine can also act as a client machine.

Client machines use a modified kernel that maintains contact with the File Exporter and server processes running on server machines. This collection of kernel modifications on a client machine is known as the *Cache Manager*. The main duty

of the Cache Manager is to translate file requests made by application programs on a client machine into Remote Procedure Calls (RPCs) to File Exporter processes on File Server machines.

When the Cache Manager receives requested data from a File Exporter, it caches the data (stores it on disk or in memory) before passing it to the application program that requested it. In addition, DFS ensures that the Cache Manager always has access to the most current copy of the data. If the central copy of the file containing the data changes, the Cache Manager retrieves the newer version of the file the next time data from the file is requested (or in the case of read-only data, within a configurable period of time). The user does not have to direct the Cache Manager to keep a current copy; the Cache Manager's actions are automatic and completely transparent to the user.

DFS Data Access Management

To synchronize distributed access to data, the File Exporter on each File Server machine distributes *tokens* to clients that access data from the machine. The File Exporter uses tokens to manage access to data and metadata. Tokens guarantee that each client is working with the most recent version of the data and that multiple clients are not accessing the same data in a conflicting manner. Tokens are fully transparent to both users and administrators.

When a client such as the Cache Manager needs to access or change a file or directory that is managed by the File Exporter, it first requests the appropriate tokens for the data from the File Exporter. The File Exporter's response to the client's request depends on the data the client wants to manipulate, the operation the client wants to perform on the data, and whether any other clients currently have tokens for the data.

If no other clients have tokens for the data, the File Exporter can issue the client the appropriate tokens. If outstanding tokens for the data exist, the File Exporter can grant the request (if no conflicts arise between the request and the outstanding tokens), revoke the existing tokens to grant the request, or consider the request pending until it can grant it. In some cases, the File Exporter simply refuses to grant the request. If the File Exporter gives the client the necessary tokens, the client in turn can access the data from the File Exporter in the fashion requested.

DFS Administrative Domains

In DCE, the cell is the basic unit of operation. A cell consists of from one to several thousand systems sharing an administratively independent installation of server and client machines, a unified DCE Cell Directory Service (CDS) naming environment, and a common authentication server and database. Multiple cells can exist at one geographical location. It is also possible for DFS machines at geographically distant locations to belong to the same cell. However, a machine can belong to only one cell at one time.

A user can have access to several cells. However, the user's Universal Unique Identifier (UUID) appears in the registry for only a single cell. This cell is said to be the local cell (or home cell) for the user. All other cells are considered foreign cells from the perspectives of both the user and any machines in the user's home cell.

When logging into a machine, the user authenticates to the cell to which that machine belongs. If the machine belongs to the user's home cell, the user's UUID

appears in the registry in that cell. If the machine is in a foreign cell, the user's UUID does not appear in the cell's registry; mutual trust must exist between the foreign cell and the user's home cell for the user to successfully authenticate to the foreign cell. The system administrator who configures your cell determines whether your cell participates in the global naming service. If your cell participates in the global naming service, you can permit users from foreign cells that also participate in the global naming service and that have established mutual trust with your cell to access your data, and vice versa.

DFS further extends the concept of a DCE cell by providing DFS administrative domains. An administrative domain is a collection of associated server machines from the same cell configured for administration as a single unit. A cell can include a large number of machines; administrative domains provide a means of simplifying the administration of many DFS machines in a single DCE cell by organizing a subset of the cell's machines into smaller administrative units. In addition to simplifying the management of DFS in a DCE cell, administrative domains bring fine levels of granularity and flexibility to DFS administration in general.

A cell can have one or more administrative domains. An administrative domain, like a cell, can include server machines that perform many of the machine roles mentioned previously. A machine can be a member of multiple domains, but all of the machines in a domain must be members of the same cell. For example, all of the domains in a cell can use the same Binary Distribution machine for a machine type, but that machine must be in the same cell as all of the machines in all of the domains. Administrative domains are transparent from the end-user's perspective.

DFS Administrative Lists and Groups

Administrative lists are files that are used with administrative domains to determine which individuals are allowed to issue commands that affect specific processes and data. Being a member of an administrative list is analogous to having the permissions necessary to issue requests to the associated server process. Individual users can be placed on administrative lists to grant them the administrative privileges associated with the lists. Groups of users can also be placed on administrative lists to grant the privileges associated with that list to all of the members of the group simultaneously; the members of a group have all the privileges associated with any administrative lists in which the group is included. In addition, server machines can, and in some cases must, be placed on administrative lists.

You can grant users administrative privileges by adding them to different administrative groups. You do not need to explicitly grant the individual users all of the privileges associated with each group. You can then modify the group's privileges rather than the privileges of each of its individual members.

For instance, you can create a group called **domain1.admin** and include it in the administrative lists necessary to allow its members to administer data on the File Server machines in a single domain. You can then assign users to the **domain1.admin** group to grant them administrative privileges on the File Server machines in the domain; you do not need to include each individual user in all of the necessary administrative lists in the domain.

Similarly, you can create additional groups for other administrative tasks, such as managing processes or installing new system binaries, and include the same or different users in these groups. Users have only the privileges associated with the administrative lists in which they are included. Unless users are also members of

other administrative lists in a domain or in the cell to which a domain belongs, their membership in an administrative list on a machine grants them no additional privileges beyond the scope of that administrative list. You can limit a group's administrative duties by placing it on only certain administrative lists in a domain.

The documentation in this part of the guide frequently states that the user who is to perform a task must be included in the appropriate administrative lists. Users can be included directly, by having their usernames included in the list, or they can be included indirectly, by being assigned to a group that is included in the list; either method is sufficient.

Administrative lists are only one form of security used in DFS. As the next section describes, DCE Access Control Lists (ACLs) are also used to limit access to files and directories. Many DFS operations require that the issuer be included on the proper administrative lists *and* have the proper ACL permissions.

DCE Local File System

The DCE Local File System (DCE LFS) is a high-performance, log-based file system. DCE LFS supports the use of aggregates. A DCE LFS aggregate is physically equivalent to a standard UNIX disk partition, but it also contains specialized metadata about the structure and location of information on the aggregate. DCE LFS maintains a log of all modifications made to the metadata by operations such as file creation and modification. The log is completely transparent to users and requires no special administration. In the event of an abnormal system shutdown, DCE LFS replays the logged information about the metadata and uses it to return the aggregate to a consistent state.

To further ensure file system consistency after an abnormal shutdown, DFS also includes the DFS Salvager. The Salvager is used to return consistency to a file system when the file system has structural problems that cannot be corrected automatically by replaying the log or when the DCE LFS log is damaged. To detect and repair inconsistencies in the file system that the log mechanism cannot repair, the Salvager reads and analyzes structural and organizational information about the file system. The DFS log mechanism and Salvager are analogous in many respects to an enhanced **fsck** program, the mechanism commonly used to return consistency to other file systems. One difference between the two is that the **fsck** program is commonly used to check file systems whenever a machine is restarted, whereas the Salvager needs to be used to verify a file system only when log recovery fails.

DCE LFS aggregates also support the use of *filesets*. A DCE LFS fileset is a hierarchical grouping of files managed as a single unit.

Note: Beginning in AIX 4.1, the term fileset is used to refer to an AIX licensed program product. In a DFS context, a fileset is a collection of related files that are organized into a single unit. These two uses of the term are distinct; the intended meaning should be inferred from its context.

DCE LFS filesets can vary in size but are almost always smaller than a disk partition. With DCE LFS, multiple filesets can be stored on a single aggregate, providing flexible disk usage. A non-LFS partition (for example, a UNIX partition) can be exported to the namespace for use as an aggregate with DFS. However, it can store only a single fileset (file system), regardless of the amount of data actually stored in the fileset. (The terms *non-LFS aggregate* and *non-LFS fileset* are

used to refer to exported non-LFS partitions and the file systems they contain.) For AIX, this can be AIX Journaled File Systems or CD-ROM File Systems.

The unique metadata structure of DCE LFS aggregates also supports additional fileset operations not found on standard, non-LFS partitions. With DCE LFS, the potentially small size of filesets allows them to be easily managed for maximum system efficiency. Also, each DCE LFS aggregate can store multiple DCE LFS filesets. A system administrator can move filesets from one DCE LFS aggregate to another or from one machine to another for load balancing across machines. If the complete contents of a user's home directory are stored in one fileset, the entire directory moves when the fileset is moved.

Each DCE LFS fileset corresponds logically to a directory tree in the file system. Each fileset maintains, on a single DCE LFS aggregate, all of the data that makes up the files in the directory tree. For example, if you maintain a separate fileset for each user's home directory, you can keep a person's files together but separate from those of other users.

The place at which a DCE LFS fileset is attached to the global filespace is called a *mount point*. A mount point looks and acts like the root directory of the fileset. This correspondence between a directory and fileset also simplifies the process of file location. A mount point identifies a fileset by name so that DFS can automatically locate the fileset, even if the fileset is moved between aggregates or machines.

Each DCE LFS fileset has a fileset quota associated with it. A fileset's quota specifies the maximum amount of disk space the information in the fileset can occupy. Quota is set on a per-fileset basis, so it can be increased for filesets that contain more data and decreased for filesets that do not need the additional disk space.

DCE LFS does not fully expand sparse files through backup and restore operations. Sparse files are generally used by database applications and provide highly efficient use of disk space. In a sparse file, only the actual data stored in the database is physically stored on disk. Blank (or ″zero″) records are not stored, although the database correctly shows them as blank when their byte offset addresses are accessed.

In DCE LFS, a replica or backup of a sparse file remains largely sparse. The file dump copy of the sparse file is broken into 64-KB chunks, and only chunks that contain actual data physically occupy 64 KB of actual space on the disk. The 64-KB granularity is imposed for performance reasons. (Smaller granularities degrade the speed of data access.)

DCE LFS also supports the use of DCE ACLs to set permissions on directories and files in DCE LFS filesets. DCE ACLs extend the standard UNIX permissions, which are set with UNIX mode bits, to offer more precise definitions of access permissions for directories and files. ACLs and administrative lists restrict access to DFS management operations in a cell or domain to specifically authorized users.

DFS Replication

DCE LFS allows you to replicate (copy) DCE LFS filesets. When you replicate a DCE LFS fileset, you place read-only copies of it on multiple server machines. The unavailability of a single server machine housing a replicated fileset does not usually interrupt work involving that fileset because copies of the fileset are still available from other machines. The replication of commonly used configuration and

binary files on multiple server machines greatly reduces the chances of their being unavailable as the result of server machine outages. Replication also prevents a machine from becoming overburdened with requests for files from a frequently accessed DCE LFS fileset. Replication is supported only for DCE LFS filesets, not for non-LFS filesets (file systems on non-LFS partitions).

Two types of replication are available with DCE LFS: Release Replication and Scheduled Replication. With Release Replication, you issue a command to copy a source fileset to the server machines housing its read-only replicas every time you want to update the replicas to reflect the current contents of the read/write fileset. This type of replication is useful if the fileset seldom changes or if you need to closely monitor the replication process.

With Scheduled Replication, you specify replication parameters that dictate how often DFS is to automatically update replicated filesets with new versions of source filesets. This type of replication is useful if you prefer to automate the process and do not need to track exactly when releases are made. Both types of replication produce the same result: source filesets are copied to different server machines. The system administrator chooses which type of replication to use with each fileset.

Note:

Replicas of sparse files do not expand to their full size. Replicas have a minimum granularity of 64 KB; any 64 KB ″chunks″ that do not contain actual data require no storage space. Chunks that contain data expand to occupy a full 64 KB of storage space.

DFS Backup System

DFS provides two methods of managing backups: the DFS Backup System and backup filesets. With the DFS Backup System, you can copy data from filesets to tape and restore the data from tape in the event that the data is lost. Information about backups and tapes is maintained in the Backup Database. The database itself can be copied to tape and restored in the event of its corruption. Backups of DCE LFS filesets are supported.

You can perform both full and incremental backups, or dumps. A full backup copies all of the data in a fileset to tape; an incremental backup copies only those files that have changed since the last full backup to tape. A backup schedule, or dump hierarchy, records the specified filesets to be included in a backup.

You can restore data from tape in the same manner. A full restore re-creates the data as it was at its last backup, including any changes from the last full backup and any subsequent incremental backups; a date-specific restore re-creates the data as it was at a specific point in time, including data from any incremental backups done before the specified date. You can restore individual filesets or an entire aggregate.

The DFS Backup System supports automated backup devices, such as jukeboxes and stackers. By specifying parameters in a configuration file and writing the appropriate executable routines, you can enable the DFS Backup System to change tapes, select tapes, and handle errors.

Note:

Sparse files, when copied through the DFS Backup System, do not expand to their full size. However, the sparse file copy has a minimum granularity of 64 KB; any ″chunk″ that contains no actual data requires no storage space on the tape.

Backup filesets capture the state of source data at the time the backup is made; they do not involve the Backup System. You can create a backup version of a user's DCE LFS fileset and mount it as a subdirectory of the user's home directory, naming it something appropriate such as **.OldFiles** or **.BackUp**. The user can then, without assistance, restore to a read/write fileset any files deleted or changed since the backup fileset was made. Users cannot change the data in their backup filesets, but they can copy the data to a regular directory in a working, read/write fileset and use it there.

DFS Database Distribution

DFS houses fileset and Backup System information in two administrative databases. The Fileset Location Database (FLDB) stores information about the locations of filesets; the Backup Database records information about backups and tapes. To maintain file system reliability and availability, the two databases are replicated on multiple server machines. If any of the machines housing a database becomes unavailable, the database is still available from other machines.

Administrators can use the multihomed server capabilities in DFS to provide the most efficient network access from DFS clients to FLDB server machines. Each machine can have up to four IP addresses, providing network connections to the subnetworks or networks that have the highest concentration of DFS clients. Should a particular FLDB machine connection become unavailable, the Cache Managers on the various DFS clients then reference their lists of server preferences to connect to the next ″preferred″ address for an FLDB machine. By default, the preference values are chosen to make reasonable decisions about the order in which servers are accessed. For example, the default preference values bias a Cache Manager to first access FLDB machines within its same subnetwork before contacting machines in other subnetworks.

To synchronize the information in the databases, DFS uses a library of utilities called *Ubik*. Ubik is a synchronization mechanism that distributes changes to fileset and backup information to all copies of the appropriate database. Administrators need to be aware of which machines store copies of a database only when the machines are configured. Once the machines are configured, administrators, like users, never need to know which server machines store copies of a database; they merely make changes to information in a database, and Ubik coordinates the updating of the information to all sites at which the database is replicated. The distribution across the database sites is automatic and almost instantaneous.

The DFS scout Program

DFS also includes the **scout** program, which system administrators can invoke from a single client machine to monitor the File Exporters on many File Server machines at one time. The **scout** program uses a graphical display to present machine usage statistics about the File Exporters it is monitoring. It can be instructed to highlight any value that exceeds a specified threshold for a statistic it is monitoring. It also indicates any machine whose File Exporter fails to respond to requests for information.

The **scout** program tracks the following statistics about the File Exporter on each machine being monitored: the number of connections that principals (users and machines) have open to the File Exporter, the number of fetches (requests to send data) the File Exporter has serviced from clients, the number of stores (requests to store data) the File Exporter has accepted from clients, the number of client machines that have communicated with the File Exporter, and the number of kilobytes available on aggregates on the File Server machine on which the File Exporter is running.

Access to DFS from NFS

The NFS/DFS Authenticating Gateway provides authenticated access to DFS by means of the Network File System (NFS). The NFS/DFS Authenticating Gateway allows users of NFS clients to obtain DCE credentials, which they can use for authenticated access to data in the DFS filespace. Without the NFS/DFS Authenticating Gateway, users of NFS clients have only unauthenticated access to data in the DFS filespace.

To use the NFS/DFS Authenticating Gateway, configure a DFS client as a Gateway Server, and export the root of the DCE namespace from that client. Then mount the DCE namespace on each NFS client from which DFS access is desired. Users who have DCE accounts can then access DFS from the NFS clients; doing the appropriate authentication to DCE provides these users the privileges and permissions associated with their DCE identities. Mounting the DCE namespace also provides unauthenticated DFS access to users who do not have DCE accounts. See the *IBM DCE for AIX, Version 2.2: NFS/DFS Authenticating Gateway* for information about configuring and using the NFS/DFS Authenticating Gateway.

Advantages of DFS

The components and features of DFS provide many advantages over nondistributed file systems and other file systems in general. The following subsections briefly describe some of the advantages available with DFS but not typically available with other file systems.

Faster Restarts and Better Reliability

Restarting DFS after an abnormal system shutdown is faster if DCE LFS is used because DCE LFS logs information about operations that affect the metadata associated with DCE LFS aggregates and filesets. When the system is restarted, DCE LFS replays the log to reconstruct the metadata. It returns the system to a consistent state faster than non-LFS file systems that must run the **fsck** command.

Access to information is more reliable in DFS for a number of reasons (in addition to the logged metadata already mentioned). In a distributed file system, multiple clients such as the Cache Manager can attempt to access the same data simultaneously. DFS uses tokens to ensure that users are always working with the most recent copy of a file and to track who is currently working with the file. Tokens identify operations the client can perform on the data. They also act as a promise from the File Exporter that it will notify the client if the centrally stored copy of the data changes; following such notification, the client can then retrieve the most recent copy of the data the next time it is requested by a user.

DFS also improves the reliability of data access by allowing you to replicate commonly used DCE LFS filesets on multiple File Server machines. When you

replicate a fileset, you place an identical copy of the fileset on a different File Server machine. The unavailability of a single server that houses the fileset generally does not interrupt work involving that fileset because the fileset is still available from other machines.

Better Recovery from Failure

As detailed previously, recovering from an abnormal system shutdown is easier because DCE LFS automatically maintains a log of the current state of the metadata associated with aggregates and filesets. Recovery from more severe system failures that can include the loss of data is also easier because the DFS Backup System allows system and user data to be backed up to tape. Information about backups, which is maintained in the Backup Database, can be used to easily and reliably restore system and user data to its state at the last backup or at a specific date.

Recovery from system failure in most UNIX file systems involves using the **fsck** command to ensure that no file systems are corrupted and, if they are, to correct the problems so that they do not spread through the entire file system. In DFS, such measures are not required at every restart. When they are needed, they involve the use of the DFS Salvager to locate and correct serious data corruption from which DCE LFS cannot recover without assistance. In some cases, problems may occur in the basic structure of the file system or the log may be damaged. The Salvager lets you check the file system and correct problems to prevent corruption of the entire DCE LFS aggregate on which the file system is stored; it detects and repairs inconsistencies in the file system's metadata to return the file system to a consistent state.

After a File Server machine is restarted, the File Exporter attempts to restore consistent access to data on the machine. For a brief time after the restart, it prevents all clients from establishing new tokens for data on the server machine. During this recovery period, it honors requests only to reestablish tokens from the clients that held them before it was restarted; these clients have the opportunity to recover their tokens before any client can request conflicting tokens. Providing clients with the opportunity to regain their tokens after a File Server machine restart is one form of a practice referred to as *token state recovery*.

Improved File Availability, Access Time, and Network Efficiency

Increased file availability and network efficiency in DFS is provided through three mechanisms: replication, caching, and multihomed file servers.

- Replication increases file availability by allowing DCE LFS filesets to be reproduced on multiple server machines, which minimizes the effects of machine outages. If one machine housing a read-only copy of a DCE LFS fileset is unavailable, other replicas of the fileset are usually available from other machines.
- Locally caching data decreases access time to the data. The cache is an area of a client machine's local disk or memory dedicated to temporary data storage. Once data is cached, subsequent access to it is fast because the client machine does not need to send a request for it across the network. Thus, caching also minimizes network traffic. As noted previously, DFS ensures that each client housing cached read/write data always has access to the most recent version of the data; in the case of cached read-only data, DFS ensures that each client has access to the most recent version of the data within a configurable period of time.

- Multihomed File Servers both help administrators make efficient use of their networks and increase file availability. Network efficiency is improved by allowing administrators to create connections between file servers and the subnetworks or networks wherein most of the DFS clients reside. Multiple network connections per File Server also increases file availability in that a fault in one section of the network is less likely to make a File Server unavailable.

Efficient Load Balancing and File Location Transparency

Load balancing of data is more efficient in DFS than in standard nondistributed file systems. One reason is the use of replication, which allows DCE LFS filesets to be reproduced on multiple machines. Requests for files from frequently used DCE LFS filesets are then spread across different machines, preventing any one machine from becoming overburdened with data requests. Multihomed server capability makes it possible for each machine to have multiple connections to the network, allowing direct connections to the subnetworks that provide the most requests. These connections help reduce cross-router traffic within the network.

Fileset characteristics in DCE LFS also improve load balancing. DCE LFS filesets are typically smaller than standard UNIX and other non-LFS filesets; DCE LFS aggregates can accommodate multiple DCE LFS filesets for flexible disk usage; and DCE LFS filesets can be moved between aggregates on different File Server machines. The ability to store multiple filesets on a single aggregate is integral to being able to move filesets in DFS.

DFS automatically tracks every fileset's location, even when the fileset is moved between aggregates or machines. The location of any fileset is automatically maintained in DFS by the Fileset Location Database (FLDB). This database tracks the machine and aggregate that houses each exported fileset (DCE LFS or non-LFS). Therefore, the user never needs to know the machine or aggregate that actually houses the fileset.

The FLDB relieves the system administrator of the burden of manually tracking each fileset's location, thus freeing the administrator to concentrate on more important administrative duties. Also, the master version of the database is typically replicated and synchronized (using Ubik) on multiple server machines, making access to the FLDB more reliable.

Extended Permissions

DFS extends the UNIX permissions to provide a more precise definition of access permissions for directories and files. UNIX defines three access permissions: read (**r**), write (**w**), and execute (**x**). DCE ACLs define six permissions: the UNIX read (**r**), write (**w**), and execute (**x**) permissions and additional control (**c**), insert (**i**), and delete (**d**) permissions. You can grant any of the six available permissions for a directory. You can effectively grant only the read, write, execute, and control permissions for a file.

Depending on an object's type (directory or file), you can assign it different types of ACLs. Directories are referred to as *container objects*; they can be assigned Object ACLs (which control access to the object), Initial Container Creation ACLs (which provide default ACLs for newly created subdirectories), and Initial Object Creation ACLs (which provide default ACLs for newly created files the directory contains). Files are referred to as *simple objects*; they have only Object ACLs.

Increased Interoperability and Scalability

Data from non-LFS file systems can be used with DFS. You can export a non-LFS disk partition to the DCE namespace for use as an aggregate in DCE. Although it can be accessed in the namespace, an exported partition still holds only the single file system it contained when it was exported. Additionally, a non-LFS aggregate does not necessarily support features such as logged information about metadata, DCE ACLs, and fileset replication, which are available with DCE LFS.

In DFS, the Basic OverSeer Server (BOS Server) monitors DFS processes on server machines. Once it is started and configured, the BOS Server continues to monitor other DFS server processes with minimal intervention from the system administrator. Decreased administrative obligations, coupled with high performance and a high client-to-server ratio, make DFS a scalable system. Server and client machines can be added to a DFS configuration with little impact on other servers or clients and with few additional administrative responsibilities.

Increased Security and Administrative Flexibility

DFS supports enhanced administration and security by making DCE ACLs available with objects in DCE LFS filesets and by using administrative lists with DFS server processes. In addition, you can place groups of users on ACLs or administrative lists to extend the same permissions or privileges to multiple users simultaneously. Because each server process on a server machine has its own administrative list, a fine granularity of control with respect to server process administration is possible.

In DFS, you can enable or disable the honoring of **setuid** and **setgid** programs on a per-fileset and per-Cache-Manager basis. Thus, you can direct a specific Cache Manager to enable **setuid** and **setgid** programs located in a specific fileset (such as one that stores system binary files).

DFS allows you to set the RPC authentication levels for Cache Manager to File Server communications. These levels can be set individually for each Cache Manager and File Server. In addition, you can also set advisory RPC authentication bounds on a per-fileset basis. Although not currently enforced, the advisory bounds serve to bias the Cache Manager's selection of an initial RPC authentication level.

Consistency of Configuration and Binary Files

In DFS, designated machines can be used to store central copies of system configuration and binary files. The files can then be distributed from these machines to the other machines that use them, thus ensuring consistency among the various server machines in the network.

In DFS, two types of machines are responsible for housing common configuration and binary files: System Control machines and Binary Distribution machines. A single instance of the System Control machine distributes all of the common configuration files such as administrative lists to all of the machines in its domain. One Binary Distribution machine exists for each CPU/operating system type found in a cell; each Binary Distribution machine distributes the binary files for its CPU/OS type to all of the other machines of the same type in its cell.

The DFS Update Server process distributes common files from System Control and Binary Distribution machines. Server machines that rely on System Control and Binary Distribution machines for configuration and binary files run the client portion

of the Update Server (the **upclient** process). The System Control and Binary Distribution machines run the server portion of the Update Server (the **upserver** process).

Each instance of the **upclient** process frequently checks with the appropriate **upserver** process to make sure its copies of the proper files are current. If newer versions of the configuration or binary files exist, the **upclient** process copies them via the **upserver** process and installs them.

Backup Versions of Data

System and user data can be backed up to tape and restored as necessary. As mentioned previously, the DFS Backup System enables data from filesets to be backed up to tape. In the event of disk corruption or similar problems, the data can be restored from tape to the file system.

In addition, backup versions of DCE LFS filesets can be created and made available via users' directories for access by users if they mistakenly lose data. This allows users to access prior versions of their files easily, without burdening system administrators with requests for assistance. The administrators are then free to focus on more critical duties.

System Monitoring

DFS provides three types of system monitoring. The first type of monitoring involves the BOS Server, which continually monitors and restarts (as necessary) all indicated DFS server processes running on a server machine. The system administrator indicates the processes the BOS Server is to monitor. The BOS Server restarts itself and all other indicated server processes on the machine once a week (to use new binary files, for example); it also checks each specified process once a day to ensure that each process is using the most current binaries. Once it is running on a machine, the BOS Server requires little intervention from the system administrator.

The second type of monitoring involves the **scout** program, which system administrators can use to monitor File Server machine usage. This program allows administrators to determine which machines and aggregates are experiencing the most data requests, as well as which machines are functioning properly. An administrator can use the **scout** program to track the File Exporters on many File Server machines from a single client machine. The **scout** program presents statistics about the File Server machines and File Exporters it is monitoring in a graphical format, and it allows the system administrator to set attention thresholds for the statistics being monitored.

The third type of monitoring involves the **dfstrace** command suite, which system administrators can use to trace DFS processes that run in either the user-space or the kernel. Commands in the suite allow sophisticated administrators and system developers to obtain internal tracing information that they can use to diagnose and debug system problems. Because **dfstrace** commands are beyond the scope of normal DFS administration, information about them is presented only in "Chapter 11. Monitoring and Tracing Tools" on page 299.

Interaction with Other DCE Components

Because DFS is built on top of other DCE components, an understanding of those other components is essential for an understanding of DFS. The information in this section is intended only as an overview of some of those other components; it is assumed the reader has read the *IBM DCE for AIX, Version 2.2: Introduction to DCE* and understands the following DCE components:

- The DCE Security Service, especially the keytab file and ACLs. (See the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for information about using keytab files and for complete details about setting ACLs.)
- The DCE Directory Service, especially details about the namespace. (See the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for complete details about configuring and using namespace components.)
- The DCE Distributed Time Service, especially client and server machine synchronization. (See the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for complete details about configuring and using the Distributed Time Service.)
- The DCE Remote Procedure Call Facility, especially client and server machine communications. (See the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for complete details about configuring your machines to use RPCs.)

Many of these services also interact with each other. (See the *IBM DCE for AIX, Version 2.2: Introduction to DCE* and the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for more information on these components and their mutual interaction.)

DCE Security Service

The DCE Security Service is composed of three primary services: the Authentication Service, the Registry Service, and the Privilege Service.

- The DCE Authentication Service component performs several security functions that interact with DFS. It ensures that only certified users can log into and use the system, and it ensures that only authorized machines can communicate with other machines in the network.
- The DCE Registry Service maintains a Registry Database. This database contains information similar to that stored in UNIX password files, such as users, groups, and account information. An account defines who can log into the system and includes information about passwords and home directories.
- The DCE Privilege Service component ensures that those who are using the system have the necessary permissions to perform the operations they request.

These three services rely on the DCE Security Server, the **secd** process.

The DCE Security Service also includes the following two facilities:

- The DCE Access Control List Facility provides an interface that allows users to set different levels of protection on file system objects such as directories and files. Users can grant permissions to individuals, or they can define groups of users and grant permissions to the groups. They can then add individuals to a group to grant them the same permissions as the group, or they can remove individuals from a group to restrict their permissions. An object's ACLs interact with the protections provided by the object's UNIX mode bits.

- The DCE Login Facility initializes a user's security environment in DCE. It employs a user's password to authenticate the user to the DCE Security Service, returning authentication information associated with the user. This information is used to authenticate the user to other distributed services such as those in DFS.

Note: If you are using DFS, you must use **dce_login** after the DFS client is configured and running on your machine in order to have authenticated access to DFS objects. Credentials acquired before DFS is running is not recognized by DFS.

DCE Directory Service

The DCE Directory Service provides a consistent way to locate resources such as machines and other services anywhere in a networked computing environment. The DCE Directory Service has three main components:

- The Cell Directory Service (CDS) manages names within a cell. Each resource has a CDS entry that is unique within its local cell.
- The Global Directory Agent (GDA) is a ″gateway″ between the local and global naming environments. It supports cell interoperability by allowing CDS to access a name in another cell via either GDS or the Domain Name System (DNS), a widely used global naming environment.

Examples of CDS Entries

Examples of CDS entries in both GDS and DNS global naming formats follow. The first example shows a CDS entry for a server machine in DNS format:

```
/.../abc.com/hosts/fs1/self
```

The second example shows a similar entry in GDS format:

```
/.../C=US/O=abc/OU=Writers/hosts/fs1/self
```

In addition to their global names, all CDS entries have a cell-relative name, or local name, that is usable only within the cell where the entry exists. The cell-relative name begins with the **/.:** prefix, which replaces the global cell name. An example of a CDS entry that uses the cell-relative prefix follows:

```
/.:/hosts/fs1/self
```

DFS Filespace

The default name for the root of a DCE cell's DFS filespace is **fs**, which is an entry in the cell's namespace. The **fs** entry, referred to as a *junction*, serves as a boundary between the CDS namespace and the DFS filespace. The contents of the **fs** junction provide the information necessary to access files and directories in the filespace.

The name **fs** is only a default; it is not considered to be well known and, thus, can vary from cell to cell.

The name of a file or directory object in DFS includes the **fs** element in its pathname to indicate that the object resides in the DFS filespace. Entries in the DFS filespace can be represented in DNS, GDS, and cell-relative format. The following examples are valid ways to refer to a directory in the **abc.com** cell, which uses DNS:

```
/.../abc.com/fs/usr/terry
/.:/fs/usr/terry
```

The following examples are valid ways to refer to a similar directory in the **def.com** cell, which uses GDS:

```
/.../C=US/O=def/OU=Writers/fs/usr/dale
/.:/fs/usr/dale
```

Entries in the DFS filespace also have a DFS-relative name that, like the cell-relative prefix, is usable only within the cell in which the entry exists. The DFS-relative name begins with the **/:** prefix, which is an abbreviation for both the global cell name and the **fs** entry that begins the DFS filespace. An example of a directory name represented in DFS-relative format follows; the name is valid only from within the local cell.

```
/:/usr/terry
```

Commands that use CDS interfaces know how to interpret the **/:** prefix. For example, the **dcecp rpcentry** command is able to interpret the **/:** prefix as ***/.../cellname*/fs**.

However, commands that access file and directory objects in the DFS filespace rely on the presence of a symbolic link to resolve the **/:** prefix. The symbolic link **/:** must reside in the root directory of the local machine, and it must point to the location of the DFS junction in the CDS namespace of the local cell. The link must be created on each DFS client machine; it is usually created when a machine is configured as a DFS client.

For example, suppose a pathname that begins with the **/:** prefix is used with the **dcecp acl** command. For the command to succeed, the symbolic link **/:** must exist in the root directory of the machine on which the command is issued, and the link must point to the CDS entry ***/.../ cellname*/fs** (or whatever name is used for the DFS junction in the local cell); otherwise, the command fails because it cannot interpret the **/:** prefix.

Note that the **/.:** and **/:** prefixes are abbreviations intended primarily for interactive use, not for use in persistent storage such as shell scripts. Use global names (of the form ***/.../ cellname***) for pathnames in persistent storage. Note especially that the **/.:** and **/:** prefixes cannot be used in strings in which a **:** (colon) has a reserved meaning. For example, you cannot use the prefixes in the definition of a PATH environment variable in operating systems such as the UNIX system; in this case, the **:** is used to separate different pathnames, so including a prefix in the definition of a PATH environment variable violates the reserved nature of the **:** for that variable.

Note: Examples and output in this part are displayed in DNS format. Use whatever format is appropriate for your cell (DNS or GDS); if it is enabled in your cell, a cell-relative prefix (**/.:**) or DFS-relative prefix (**/:**) can be substituted wherever a path begins with **/.../abc.com** or **/.../abc.com/fs**. Also, the term *DCE pathname* refers to a name specified in any acceptable DCE Directory Service format. Finally, the examples in this part use the default, **fs**, as the junction of the DFS filespace.

DCE Distributed Time Service

The DCE Distributed Time Service (DTS) provides precise synchronization for system clocks in a network. In DFS, clock synchronization is important for communications between client machines using the Cache Manager and server

machines running the File Exporter and other server processes. Clients and servers must refer to a common time standard for communications to remain constant and for data to remain available.

For example, each client that obtains tokens from a File Exporter has a lifetime with respect to that File Exporter. The client must renew its lifetime before it expires to ensure that its tokens are not revoked without its knowledge. If the client and File Exporter disagree on the current time, the File Exporter may believe the client's lifetime has expired before the client does. In this case, the File Exporter may revoke the client's tokens without its knowledge.

Clock synchronization is also important for replicated Fileset Location Databases and Backup Databases, which must be coordinated on different server machines. Machines that house replicated databases must remain in constant contact to ensure that each server has the current copy of the database. If the machines disagree on the time, they may believe they are no longer in touch with each other, in which case they can refuse all requests for information. Synchronization problems of this nature can result in unnecessary disruption of database access.

DCE Remote Procedure Call

The DCE Remote Procedure Call (RPC) facility provides communications between client and server machines in a network. For the Cache Manager on a client machine to send a request for data or other resources to a server machine, it must know how to locate the File Server machine in the network and how to communicate with it. An RPC requires the use of a binding handle for the File Server machine on which a fileset resides.

The binding handle includes the server machine's network address, an identifier for the protocol used to communicate with the machine, and an endpoint (often a port number) for communications with the machine. It also contains user authentication information about a user who requests data. The Cache Manager uses this binding handle to communicate with the server machine.

The same process is used to effect communications between different server machines. For example, DFS employs Ubik to synchronize copies of the Fileset Location Database and Backup Database. Instances of Ubik that coordinate the databases on different server machines rely on RPCs to communicate with each other. Communication failures resulting from RPC problems can cause unnecessary disruption of database access.

When a server process first starts, it registers its process endpoint with the endpoint mapper service of the **dced** process. The **dced** process running on a server machine provides the remote location information required by clients to communicate with server processes running on the server machine. The **dcecp** program allows system administrators to manage the **dced** process on a machine. Many RPC administrative tasks, however, are performed automatically when a server first starts.

The UUID Facilities are another component of the DCE RPC employed by DFS and other DCE components. The commands and routines in the facilities are used to generate Universal Unique Identifiers (UUIDs). These UUIDs are used to uniquely identify resources such as machines and processes. For example, the Backup System uses UUIDs to identity the Tape Coordinator processes on machines that are used to back up data to tape.

System Administration: A Task Overview

The administration of DFS can be divided into three general types of tasks:

- Fileset management: efficiently organizing the filesets in your cell and maintaining appropriate backup versions of filesets that contain binary and user files
- System management and configuration: monitoring the performance of the file system software and making adjustments as necessary
- Security issues: establishing the correct procedures and policies to ensure the security of the file system

DFS provides the commands introduced in this section to help you with these tasks and procedures. Most DFS commands are divided into the following categories, or command suites:

bak The **bak** command suite is used to copy files from the file system to a backup tape and to restore them from tape to the file system, as necessary. System administrators issue **bak** commands to operate the DFS Backup System; users do not use them.

bos The **bos** command suite is used to contact the Basic OverSeer Server (BOS Server), which is used to monitor and alter DFS processes on server machines in a cell. System administrators issue **bos** commands to monitor and control server processes and security; users do not use them.

cm The **cm** command suite is used to customize the Cache Manager, which runs in the kernel on client machines. System administrators issue **cm** commands to modify RPC authentication levels for communications with File Servers and to set **setuid** and device file status; users employ them to check machine and cell status and to determine machine and cache information.

dfstrace

The **dfstrace** command suite is used to trace DFS processes to obtain debugging information. System administrators use **dfstrace** commands to help diagnose DFS problems; users do not use them. The **dfstrace** commands are provided for knowledgeable administrators and developers; information about the commands is provided only in “Chapter 11. Monitoring and Tracing Tools” on page 299.

fts The **fts** command suite is used to manage system and user filesets. System administrators issue **fts** commands to create, move, replicate, remove, and set advisory RPC authentication bounds for filesets; users employ them to check fileset quota information.

DFS also includes a number of miscellaneous, nonsuite commands; for example, the **scout** command is used by system administrators to monitor File Exporter usage statistics. Tthe NFS/DFS Authenticating Gateway also includes an additional command, **dfsiauth**, to administer DCE credentials for NFS users.

System administrators can issue all DFS commands; users can issue only those DFS commands that require no administrative privileges (for example, the **fts lsquota** command). “DFS Command Structure and Help” on page 23 provides information about the structure of DFS commands and describes how to receive online help for them.

Refer to the Reference part of this guide and reference for detailed discussions of the various DFS commands. Refer to the *IBM DCE for AIX, Version 2.2:*

Administration Commands Reference for complete details about the security commands mentioned in this section. Consult the remainder of the chapters in this guide for information about fileset management, system management, and most security issues referred to in this section.

Note: DCE 2.2 for AIX provides additional commands to perform initial DFS configuration tasks and some DFS administrative tasks. For specific information about the **config.dfs**, **mkbutc.dfs**, **mkfilesys.dfs**, **rmfilesys.dfs**, **rmbutc.dfs**, and **unconfig.dfs** commands, see the Reference part of this guide and reference. For more information about using the AIX SMIT utility to configure and manage DFS, see *IBM DCE for AIX, Version 2.2: Quick Beginnings*.

Fileset Management Commands

Commands in the **fts** and **bak** suites are available to help you manage system and user filesets in your cell.

Fileset (fts) Commands

You can use **fts** commands to perform the following types of tasks:

- Create a read/write fileset with the **fts create** command; mount the fileset in the file tree with the **fts crmount** command.
- Examine a mount point with the **fts lsmount** command; delete a mount point with the **fts delmount** command.
- Create a backup version of a single fileset with the **fts clone** command; create backup versions of many filesets at once with the **fts clonesys** command.
- Prepare to replicate a fileset by assigning replication parameters with the **fts setrepinfo** command.
- Define replication sites with the **fts addsite** command; remove replication sites and read-only replicas at the sites with the **fts rmsite** command.
- Create read-only replicas of a fileset with the **fts release** and **fts update** commands.
- Check the status of the Replication Server on a File Server machine with the **fts statrepserver** command; check the status of each replica of a fileset with the **fts lsreplicas** command.
- Set a fileset's quota with the **fts setquota** command; list the quota with the **fts lsquota** command.
- List fileset header information with the **fts lsheader** command.
- List FLDB information with the **fts lsfldb** command.
- Examine fileset header information and FLDB information with the **fts lsft** command.
- Move a fileset with the **fts move** command.
- Remove a fileset with the **fts delete** command.
- Dump a fileset to a byte stream format with the **fts dump** command; restore a fileset to the file system with the **fts restore** command.
- Set advisory RPC authentication bounds on a per-fileset basis with the **fts setprotectlevels** command.

Backup (bak) Commands

You can use **bak** commands to perform the following types of tasks:

- Define a fileset family, which is used to group related filesets together for copying to tape, with the **bak addftfamily** command; define specific entries in a fileset family with the **bak addftentry** command.
- Define a backup schedule with the **bak adddump** command.
- Label a backup tape with the **bak labeltape** command.
- List information from the Backup Database with the **bak lsftfamilies**, **bak lsdumps**, and **bak lshosts** commands.
- List information from a backup tape with the **bak scantape** command.
- Perform a backup with the **bak dump** command.
- Restore individual filesets to the file system with the **bak restoreft** command; restore the contents of an entire aggregate with the **bak restoredisk** command; or restore user-defined collections of filesets with the **bak restoreftfamily** command.

System Management and Configuration Commands

Commands in the **bos** and **cm** suites are available to help you monitor and administer the overall performance of DFS on the machines in your cell. The **scout** program is also available to help you monitor the File Exporter processes running on File Server machines.

Cache Manager (cm) Commands

You can use **cm** commands to perform the following types of tasks:

- List the current cache size and type with the **cm getcachesize** command.
- Set the size of the cache with the **cm setcachesize** command.
- Force the Cache Manager to discard cached data and information about the data with the **cm flush**, **cm flushfileset**, and **cm checkfilesets** commands.
- Determine the Cache Manager's preferences for File Server machines from which to access read-only filesets with the **cm getpreferences** command.
- Set or modify the Cache Manager's preferences for one or more File Server machines from which to access read-only filesets with the **cm setpreferences** command.
- Determine whether the Cache Manager allows **setuid** programs from specific filesets to execute with **setuid** permission by using the **cm getsetuid** command.
- Allow or disallow **setuid** programs from specific filesets to execute with **setuid** permission by using the **cm setsetuid** command.
- Check the status of File Server machines with the **cm statservers** command.
- Determine the cell in which a file or directory is stored, the fileset in which it is stored, and the machine on which it resides with the **cm whereis** command.
- Determine Fileset Location Database machines for the local cell and any cells with which the Cache Manager has been in contact with the **cm lscellinfo** command.
- Modify the Cache Manager initial RPC authentication levels and lower authentication level bounds with the **cm setprotectlevels** command.
- Check the current Cache Manager initial RPC authentication levels and lower authentication level bounds with the **cm getprotectlevels** command.

Basic OverSeer (bos) Commands

You can use **bos** commands to perform the following types of security tasks:

- Create and start processes with the **bos create** command.
- List information about processes with the **bos status** command.
- Start a process that is to run indefinitely or periodically with the **bos start** command.
- Start a process that is to run temporarily with the **bos startup** command.
- Stop a process permanently with the **bos stop** command.
- Stop a process temporarily with the **bos shutdown** command.
- Install new versions of binary files with the **bos install** command.
- Use old versions of binary files with the **bos uninstall** command.
- Check the dates on existing versions of binary files with the **bos getdates** command.
- Remove old versions of binary files and server process core files with the **bos prune** command.
- List the current restart times for processes with the **bos getrestart** command.
- Set the automatic restart time for processes with the **bos setrestart** command.

The scout Program

You can use the **scout** program to monitor the following types of statistics about the File Exporter on a File Server machine:

- The number of connections that principals have open to the File Exporter
- The number of fetches (requests to send data) and stores (requests to store data) that the File Exporter has serviced
- The number of active client machines the File Exporter is serving
- The number of kilobytes in use on each aggregate on the File Server machine

When a value exceeds a threshold that you designate, the **scout** program highlights the information on the screen. In addition, if the File Exporter on a File Server machine does not respond to **scout**'s probes, **scout** automatically highlights the name of the machine, alerting you to the problem.

Security Commands and Tools

Commands in the **bos** suite are also used to manage DFS administrative privileges and security in a cell. You can use **bos** commands to perform the following types of tasks:

- List the members (users, groups, and servers) of an administrative list with the **bos lsadmin** command.
- Add a member to an administrative list with the **bos addadmin** command; remove a member from an administrative list with the **bos rmadmin** command.
- List the key version numbers and either the server encryption keys or the checksums (encrypted keys) associated with the server encryption keys in a keytab file with the **bos lskeys** command.
- Add a key to a keytab file with the **bos genkey** or **bos addkey** command; remove a key from a keytab file with **bos rmkey** command.
- Enable or disable DFS authorization checking with the **bos setauth** command.

You can use the **dcecp** command to perform the following tasks related to security:

- Verify or modify ACL permissions with the **dcecp acl** command.
- Create administrative (or user) groups with the **dcecp group create** command.

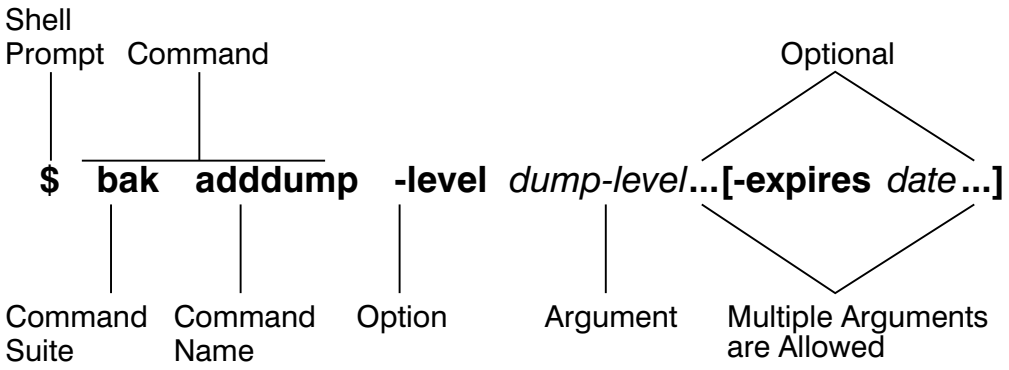
You can also use the **dfsd** command to set Cache Manager initial RPC authentication levels and lower RPC authentication level bounds. You can set the File Server upper and lower RPC authentication bounds with the **fxd** command.

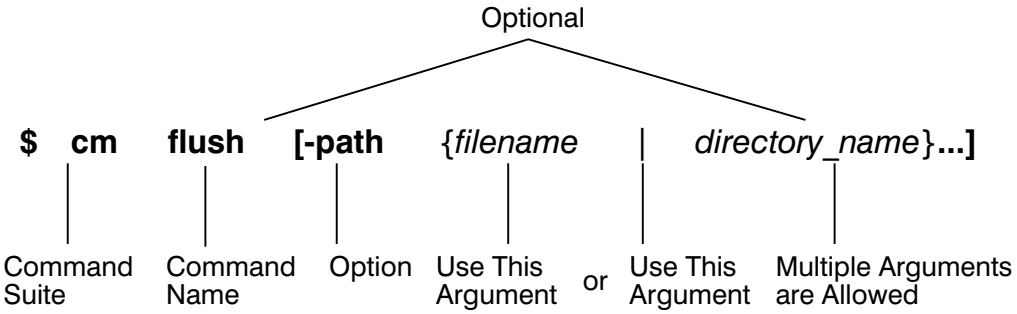
DFS Command Structure and Help

All DFS commands share a common structure. The following example shows the basic format of a DFS command:

```
$ command {-option1 argument... | -option2 {argument1 |  argument2}...} \
[-optional_information]
```

The following examples illustrate the elements of a DFS command:





The following list summarizes the elements of a DFS command:

Command
A command consists of the command suite (**bak** and **cm** in the preceding examples) and the command name (**adddump** and **flush** in the examples). The command suite and the command name must always be typed together, separated by a space. The command suite specifies the group of related commands to which the command belongs; the command name directs the server process or program to perform a specific action. Both the command suite and the command name always appear in bold font in the text.

Options
Command options always appear in bold font in the text, are always preceded by a **-** (dash), and are often followed by arguments. In the first

example, **-level** and **-expires** are options, and *dump_level* and *date* are their arguments; in the second example, **-path** is the only option.

An option and its arguments tell the server process or program which entities to manipulate when executing the command (for example, the dump level to affect and the date to assign to that level). In general, you should provide the options for a command in the order presented in the documentation.

Arguments

Arguments for options always appear in italic font in the text. The { | } (braces separated by a vertical bar) indicate that you can enter only one of two possible arguments (or use only one of two possible options). In the second example, you can enter either a *filename* or a *directory_name*; the... (ellipsis) following the closing brace indicates that multiple *filename*s, *directory_name*s, or both can be entered.

Optional Information

Some commands have optional, as well as required, options. Optional information is enclosed in [ ] (brackets). The **-expires** option and its *date* argument in the first example are optional, as are the **-path** option and its *filename* and *directory_name* arguments in the second example. Options and their arguments are optional only if they are enclosed in [ ] (brackets).

Enter each DFS command and its options and arguments on a single line followed by a carriage return at the end of the line. Use a space to separate each element (command suite, command name, options, and arguments) on a command line. Also use spaces to separate multiple arguments. Do not use a space to separate an option from its **-** (dash).

Command Shortcuts

When supplying an argument (such as a *dump_level* or *date* in the previous example), you can omit the option (such as **-level** or **-expires** in the example) associated with the argument if

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax. (The syntax for each command is presented with its description in the Reference part of this guide and reference.)
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.
- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

When two options are presented in { | } (braces separated by a vertical bar), the option associated with the first argument can be omitted if that argument is provided; however, the option associated with the second argument is required if that argument is provided.

If you must provide an option, you can abbreviate it to the shortest possible form that distinguishes it from other options of the command. For example, the **-server** option found in many DFS commands can typically be omitted or abbreviated to be simply **-s**.

You can also abbreviate a command name to the shortest form that still distinguishes it from the other command names in its suite. For example, you can shorten the **fts help** command to **fts h** because no other command names in the

fts command suite begin with the letter *h*. However, there are several **fts** commands that begin with the letter *l*, such as **fts lsquota**, **fts lsmount**, and others. To avoid ambiguity, you can abbreviate these commands to **fts lsq** and **fts lsm**; other **fts** command names that begin with *l* can be abbreviated in a similar fashion. Note that because miscellaneous DFS commands are not included in a suite, their names cannot be abbreviated.

The following example illustrates three acceptable ways to enter the same **fts lsquota** command:

1. Complete command:

   ```
   $ fts lsquota -path jlw/doc jlw/public
   ```

2. Abbreviated command name and abbreviated option:

   ```
   $ fts lsq -p jlw/doc jlw/public
   ```

3. Abbreviated command name and omitted option:

   ```
   $ fts lsq jlw/doc jlw/public
   ```

Receiving Help

You can access help for DFS commands in several ways. The following list summarizes the syntax for the different help options:

- To view the introductory page for a command suite, enter **dceman** followed by the command suite.

  $ **dceman** *command_suite*

- To view the reference page for an individual command in a suite, enter **dceman** followed by the command suite and the command name. Use an _ (underscore) to connect the command suite to the command name. Do *not* use the underscore when issuing the command in DFS.

  $ **dceman** *command_suite_command_name*

- To view a list of all commands in a command suite, enter the command suite followed by **help**.

  $ *command_suite* **help**

- To view the syntax of a specific command, enter the command suite, **help**, and the command name, in that order.

$ *command_suite* **help** *command_name*

In addition, all DFS commands include a **-help** option you can use to display the syntax of the command.

The DFS **apropos** command is similar to the UNIX **apropos** command. It displays the first line of the online help entry for any command in an indicated suite that has a specified string in its name or short description. This information is useful if you cannot remember the exact name of a command. If the string is more than a single word, surround it with " " (double quotes) or other delimiters. Enter all strings in lowercase letters.

For example, the following command produces a list of all **bos** commands with the word **create** in their names or descriptions:

```
$ bos apropos -topic create
```

All methods of obtaining help are also available with miscellaneous, nonsuite DFS commands.

Chapter 2. DFS Configuration Issues

This chapter provides summary information about the following DFS configuration issues: choosing DFS machine roles, DFS server and client configuration issues, setting up DCE LFS filesets, understanding DFS data access management, and understanding the DFS distributed database technology. Subsequent chapters provide specific details about managing DFS server machines, processes, filesets, and files.

This chapter is intended as an overview of DFS configuration issues. It also serves as a reference for the issues and considerations that go into the configuration of a cell and its administrative domains. You should read and become familiar with the information in this chapter before attempting to use any of the commands described later in this guide.

Choosing DFS Machine Roles

DFS server and client machines can run the following processes: the BOS Server to monitor other processes; the Fileset Server, Fileset Location Server, and Replication Server to manipulate DFS filesets and their replicas; the Backup Server to contact the Backup Database; the **butc** process to back up file system data to tape; and the **dfsd** process to initialize the Cache Manager on a client machine.

Each DFS server or client machine must also run the RPC, CDS, and Security processes necessary for configuration as a DCE client machine. An RPC binding must be created in CDS for the DCE pathname of each server machine, and a DFS server principal and associated account must also be created in the Registry Database for each server machine. The following sections assume that these requirements have been satisfied prior to configuring a machine as a DFS server or client machine.

The system administrator determines, at installation, which processes are to be run on which machines. A machine's role is determined by the types of processes it runs. The information in the following subsections details the different roles a machine can assume.

Each DFS server process has an associated administrative list. Users, groups, and server machines included on a process's administrative list can issue commands or calls that affect the process. Members can be added to administrative lists at any time. "Chapter 4. Using Administrative Lists and Keytab Files" on page 95 provides detailed information about the procedures used to create and maintain administrative lists. (See the Reference part of this guide and reference for complete information about the administrative privileges and permissions required to issue each DFS command.)

The Basic OverSeer (BOS) Server, or **bosserver** process, is not associated with any one machine role; it runs on every DFS server machine. Its primary function is to minimize system outages. It monitors other server processes on the local machine and restarts failed processes automatically.

By default, the BOS Server on each server machine stops and immediately restarts all DFS processes (including itself) on the machine once a week, at 4:00 a.m. on Sunday. It also checks for any newly installed binary files in the *dcelocal***/bin** directory every morning at 5:00 a.m. (Note that these restart times can be

configured.) If it finds any new files, which it does by checking for timestamps later than the time at which the corresponding process last started, it restarts the corresponding process. Because restarting processes causes a service outage, the default times are in the early morning hours, when an outage disturbs the fewest number of users. This brief suspension of services should have no effect on processes that are currently executing; the processes should continue normally once service resumes.

Install the BOS Server on all server machines to assist in administrative tasks on the machines. The **admin.bos** list is used to designate administrative users who can issue **bos** commands that affect the **bosserver** process on a server machine. Members of the **admin.bos** list can vary among different DFS administrative domains.

Overview of DFS Machine Roles

Following is a brief summary of the DFS roles a machine can assume:

- System Control machine: A single machine acts as the System Control machine for a domain, updating the other machines in the domain with identical versions of common configuration files such as administrative lists.
- Binary Distribution machine: One Binary Distribution machine of each CPU/operating system (OS) type is installed in a cell. The Binary Distribution machine updates other machines of its CPU/OS type with identical versions of system binary files.
- File Server machine: A File Server machine runs the basic set of processes necessary for storing and exporting DCE LFS and non-LFS data.
- Fileset Database machine: This type of database machine runs the process that maintains the Fileset Location Database (FLDB).
- Backup Database machine: This type of database machine runs the process that maintains the Backup Database.
- DFS client machine: Any machine can run the Cache Manager and its associated processes to act as a DFS client. This machine serves primarily as a single or multiuser workstation. It can also be configured as a Private File Server machine to export data.

Depending on the number of machines in your cell, assign the following roles to your server machines:

- In a cell with only one server machine, the machine runs all processes and fills all the necessary machine roles. Note that the System Control machine and Binary Distribution machine roles are unnecessary in this configuration.
- In a cell with two server machines, one machine acts as a Fileset Database machine and Backup Database to replicate the databases. For each database, this machine automatically assumes the role of the synchronization site and houses the source copy of the database.
- In a cell with three or more server machines, three machines run as Fileset Database machines and three machines run as Backup Database machines. This configuration allows the cell to benefit from the database replication capabilities of DFS. An odd number of database machines is best.

The software for all server processes can be installed on every server machine, even though a machine need not run every process. To change the role of a machine, simply start or stop the appropriate processes. Machine roles are not mutually exclusive; that is, any server machine can assume multiple server machine

roles, any server machine can be configured as a client machine, and any client machine can be configured as a server machine.

System Control Machines

The System Control machine in a domain stores and distributes system configuration information, such as administrative lists, shared by all DFS server machines in the domain. Configure the first server machine for any new domain as the System Control machine for that domain. It can then be used to distribute the administrative lists for that domain from its *dcelocal*/**var/dfs** directory to any subsequent server machines added to the domain.

The following processes run on a System Control machine:

- An **upserver** process (the server portion of the Update Server), which controls the distribution of common configuration files to all other server machines in the domain.
- An **upclient** process (the client portion of the Update Server), which retrieves binary files from the Binary Distribution machine of the proper CPU/OS type. (See "Binary Distribution Machines" for a description of the Binary Distribution machine.)
- A BOS Server (**bosserver** process). (See "Overview of DFS Machine Roles" on page 28 for more information about the BOS Server.)

The Update Server helps ensure that all server machines in a domain run the same version of common configuration files such as administrative lists. Configuration files are created and modified on the System Control machine, which runs the server portion, or **upserver** process, of the Update Server. Other server machines in the domain run the client portion, or **upclient** process, of the Update Server. The **upclient** processes on the other server machines in the domain frequently contact the **upserver** process on the System Control machine to verify that the most recent version of each configuration file is in use. If the most recent version of a file is not in use, the **upclient** process on each machine retrieves the most recent version from the System Control machine and installs it locally.

The server portion of the Update Server must be run on any machine that acts as a System Control machine for a domain. The **admin.up** list is used to identify all server principals that can obtain updates from the System Control machine. The list should include the names of all of the server machines in a domain.

Binary Distribution Machines

A Binary Distribution machine stores DFS binary files for processes and command suites for distribution from its *dcelocal*/**bin** and related directories to all other server machines of its CPU/OS type in a cell. Each server keeps a copy of server process binaries in a local directory; however, all the machines must be running the same version of the process for the system to perform correctly. Therefore, the binaries are installed on a single Binary Distribution machine, which acts as a source for the others. Configure one Binary Distribution machine for each CPU/OS type for which multiple machines exist in the cell.

A Binary Distribution machine runs the following processes:

- An **upserver** process (the server portion of the Update Server), which controls the distribution of binary files to other server machines of the same CPU/OS type in the cell.

- An **upclient** process (the client portion of the Update Server), which retrieves configuration files from the System Control machine.
- A BOS Server (**bosserver** process). (See “Overview of DFS Machine Roles” on page 28 for more information about the BOS Server.)

A second Update Server, different from the one used to distribute configuration files from the System Control machine, helps ensure that all server machines of the same CPU/OS type in a cell run the same binary files. Like System Control machines, Binary Distribution machines run an **upserver** process. The **upclient** processes on the other server machines of the same CPU/OS type in the cell frequently contact the **upserver** process to verify that the most recent version of each binary file is in use. If it is not, the **upclient** processes on the other server machines retrieve the most recent version from the Binary Distribution machine and install it locally. You do not have to install new software on each individual server machine because the Update Server does so automatically.

Note: On AIX machines retrieving new versions of the binaries from the Binary Distribution machine, the AIX install and update LPP history is not updated for these binaries.

The server portion of the Update Server must be run on any machine that acts as a Binary Distribution machine for a cell. The **admin.up** list associated with this Update Server is used to identify all server principals that can obtain updates from the Binary Distribution machine. The list should include the names of all machines of the same CPU/OS type in a cell.

Unless a server machine is fulfilling the roles of both System Control machine and Binary Distribution machine, different Update Servers handle the distribution of configuration and binary files. A machine configured to perform both roles runs only a single Update Server to distribute both common configuration files and system binary files.

File Server Machines

A File Server machine is used to store and export DCE LFS data for use in the global namespace. Configure enough File Server machines to contain the data to be exported from the domain. A File Server machine must run the following processes, most of which are necessary for storing filesets, exporting data, and storing replicas of filesets:

- A Fileset Server (**ftserver** process).
- The File Exporter, which is initialized by the **fxd** process, in the kernel.
- The **dfsbind** process.
- The Replication Server (**repserver** process).
- Two **upclient** processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server (**bosserver** process). (See “Choosing DFS Machine Roles” on page 27 for more information about the BOS Server.)

The Fileset Server, or **ftserver** process, provides an interface for commands that affect filesets (commands that create, delete, or move filesets, and commands that prepare filesets for archiving to tape or other media). The most common

occurrences of fileset creation and deletion are when you add or remove users from the system. Filesets are most often moved to provide load balancing among File Server machines.

The Fileset Server must run on any machine that exports data for use in the global namespace. The **admin.ft** list is used to designate administrative users who can issue **fts** commands that affect the **ftserver** process on a machine and to designate other server machines from which the machine can accept filesets. Users, groups, and machines listed in the **admin.ft** list can differ among DFS administrative domains.

The File Exporter (sometimes called the *Protocol Exporter*) runs as part of the kernel on each File Server machine. It provides the same services across the network that the local operating system provides on a local disk:

- Delivering requested files and programs to clients; storing files and programs when clients finish with them
- Maintaining the directory hierarchy structure
- Handling file-related or directory-related requests (creating, deleting, copying, and moving filesets)
- Tracking status information (including size and modification status) about each file and directory
- Creating symbolic links between files

Unlike the DFS server processes, the File Exporter is not associated with an administrative list. Instead, the command line for the **fxd** process, which is used to initialize the File Exporter and start related kernel daemons, includes an **-admingroup** option that specifies the administrative group for the File Exporter on each File Server machine. The group specified with this option must be defined in the Registry Database, as must all groups used with DFS.

Members of this administrative group can change the ACL and UNIX permissions of *all* data exported from the machine. They have the equivalent of the ACL **c** permission on all of the files and directories in each exported DCE LFS fileset, and they can effectively change the UNIX permissions on all of the files and directories in each exported non-LFS fileset. Members of the group can also change the owner and owning group of all files and directories exported from the machine. Include only highly trusted system administrators in this group.

Though similar in many respects, inclusion in the administrative group associated with the File Exporter and being logged in as **root** are *not* equivalent. A user who is logged into the local machine as **root** can perform different operations on a file or directory, depending on how he or she accesses the file or directory:

- *When accessing a file or directory via its DCE pathname*, if the user is logged into the local machine as **root** but is not authenticated to DCE, DFS treats the user as the ***/.../*** *cellname***/hosts/** *hostname***/self** principal of the local machine; in this case, the **root** user receives the permissions associated with the machine's **self** principal, which is treated as an authenticated user from the local cell. If the user is also authenticated to DCE as **root**, DFS treats the user according to the DCE identity **root**. (Note that you do not have to be logged into the local machine as **root** to be logged into DCE as **root**.)

Note: The DCE identity **root** effectively has **root** privileges for data in all exported non-LFS filesets in the cell. The identity is very powerful and represents a serious security risk. Either use the DCE **root** identity *very* cautiously or disable it altogether.

- *When accessing a file or directory via its local pathname*, the **root** user has all of the privileges commonly associated with **root**. For local access, **root** can perform any file system operation on a file or directory; for example, **root** can change the UNIX mode bits of a file or directory, change the ACL permissions of a DCE LFS file or directory, change the owner or owning group of a file or directory, or create or remove a file or directory. (A file or directory in a non-LFS fileset can always be accessed via a local pathname because a non-LFS fileset must always be mounted locally, as a file system on its File Server machine; a file or directory in a DCE LFS fileset can be accessed via a local pathname only if its fileset is mounted locally.)

Being a member of the **fxd** administrative group allows you to perform any operation on a file or directory in an exported fileset, but you may have to change the file's or directory's protections first. Being logged into the local machine as **root** lets you perform any operation on a file or directory in a locally mounted fileset immediately, without changing the protections first. Being authenticated as DCE **root** lets you perform any operation on a file or directory in an exported fileset immediately.

The File Exporter also manages the distribution of tokens to clients. It maintains an inventory of outstanding tokens, including the clients to which it has granted tokens, the data for which it has granted those tokens, and the type of each token it has granted. (A token's type dictates the operations that the client holding the token can perform on the data to which the token applies.) (See "Data Access Management in DFS" on page 52 for more information about the File Exporter's token-management mechanism.)

The **fxd** process must be run on any machine used to export data to the global namespace. (See the Reference part of this guide and reference for complete information about the **fxd** process.)

The **dfsbind** process on a File Server machine maintains user authentication information required by the File Exporter on the machine. The File Exporter uses this information to ensure that only authenticated users access data from the machine. The **dfsbind** process must be run on any machine used to export data to the global namespace.

The **dfsbind** process must also be run on all client machines. Its role on client machines is described along with client machines and their processes in "Client Machine Processes and Files" on page 38. (See the Reference part of this guide and reference for complete information about the **dfsbind** process.)

The Replication Server, or **repserver** process, manages replicas of filesets on all File Server machines. Depending on the replication method in use, you either release a new version of a fileset for distribution by the Replication Server, or the Replication Server creates replicas automatically at specified intervals. Install the Replication Server on all File Server machines, which are the machines that can store read-only replicas of filesets. No administrative list is associated with the **repserver** process.

In addition, each File Server machine must have a server entry registered in the FLDB before it can house filesets. Each File Server machine can have up to four

server entries, with each entry specifying a different host name or IP address. The server entry must exist before the **fts create** or **fts crfldbentry** command can be used to create an entry in the FLDB for a DCE LFS or non-LFS fileset from the machine. The following section discusses server entries in more detail. (See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for more information about creating server entries.)

A client machine can also be configured as a Private File Server machine to export data to the global namespace. (See “Exporting Data from a Client Machine (Private File Server Machine)” on page 35 for more information about configuring a client machine to export data.)

Fileset Database Machines

A Fileset Database machine stores the Fileset Location Database. Optimally, you should configure three or a larger, odd number of Fileset Database machines sufficient to support the File Server machines in the cell.

Each Fileset Database machine runs the following processes:

- A Fileset Location Server (**flserver** process).
- Two **upclient** processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server (**bosserver** process). (See “Overview of DFS Machine Roles” on page 28 for more information about the BOS Server.)

The Fileset Location Server (FL Server), or **flserver** process, is used to track the locations of all filesets in a cell, making file access transparent. It tracks the locations of filesets and records changes to them in the FLDB. There is one master copy of the FLDB per cell.

The first time it needs to retrieve a requested file, the Cache Manager contacts the FL Server to learn which File Server machine houses the fileset containing the file. Because of this dependency, the Cache Manager cannot retrieve a requested file if the information in the FLDB is inaccessible, even if the File Exporter on the machine that houses the fileset containing the file is working properly.

The **admin.fl** list is used to designate administrative users who can issue commands that affect the **flserver** process (operations that affect the FLDB) on a Fileset Database machine. The same **admin.fl** list should be used for all FL Servers in a cell.

A user can issue commands that affect FLDB entries for filesets on a server machine without being listed in the **admin.fl** list, provided he or she owns the machine’s server entry in the FLDB. A user gains ownership of a server entry in the FLDB by being included in the group specified as the owner of that machine’s entry with the **fts crserverentry** command. (See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for more information about creating server entries in the FLDB.)

Backup Database Machines

A Backup Database machine houses the Backup Database. As with Fileset Database machines, it is best to configure three or a larger, odd number of Backup Database machines sufficient to back up the cell’s data.

Each Backup Database machine runs the following processes:

- A Backup Server (**bakserver** process).
- Two **upclient** processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server (**bosserver** process). (See “Overview of DFS Machine Roles” on page 28 for more information about the BOS Server.)

A Backup Database machine stores the Backup Database. The Backup Database houses administrative information used in the DFS Backup System, such as the dump schedule for backups and the groups of filesets to be dumped to tape in each backup. The information in the database can be used to restore data from tape to the file system in the event of a system failure. There is one master copy of the Backup Database per cell.

The Backup Server, or **bakserver** process, maintains the Backup Database. The **bakserver** process must run on all machines that store a copy of the Backup Database. The **admin.bak** list is used to designate administrative users who can issue commands in the **bak** suite, most of which communicate with the Backup Server. The same **admin.bak** list should be used for all Backup Servers in a cell.

Commands in the **bak** suite are used to communicate with the DFS Backup System. They can be entered from any machine in the cell. Data is physically backed up and restored on a Tape Coordinator machine, which is a client or server machine that has a tape drive and runs the **butc** process to manage the drive. Information stored in the Backup Database determines the data to be backed up by a Tape Coordinator machine. (See “Chapter 9. Configuring the Backup System” on page 241 for more information on configuring and using Tape Coordinator machines.)

DFS Client Machines

A DFS client machine serves primarily as a single or multiuser workstation. It communicates with File Server machines to access files for application programs, provides local data storage, and provides computer cycles. A domain should include enough client machines to allow its users to access exported data from the local or foreign cells.

Each client machine must run

- The Cache Manager, which is initialized by the **dfsd** process, in the kernel
- The **dfsbind** process

The Cache Manager runs as part of the client machine’s kernel. It communicates with server processes running on File Server machines to fetch data on behalf of application programs. When an application program on a client machine requests data, the Cache Manager contacts the FL Server to learn the location of the fileset that houses the data. It then translates the application program’s data request into a Remote Procedure Call (RPC) to the File Exporter running on the appropriate File Server machine.

When the Cache Manager receives the requested data, it stores the data in its local cache, which is an area reserved for data storage on disk or in memory on the

client machine. It then passes the data to the application program. The Cache Manager also stores tokens it receives from the File Exporter on the File Server machine.

Within limits, the Cache Manager attempts to make the most current data available to users. The Cache Manager judges the currency of the data in its cache based on the type of fileset from which the data was retrieved:

- If the data comes from a read/write fileset, the Cache Manager uses the tokens to track the currency of the data. The cached data remains current for as long as the Cache Manager's tokens remain valid. If the read/write source of the data changes, the File Exporter revokes the tokens. The next time the data is requested, the Cache Manager retrieves the newer version to its cache before providing it to the application program.
- If the data comes from a read-only fileset, the Cache Manager compares the amount of time since the data was last verified as being current with a configurable time period associated with the fileset. If the read-only copy of the data changes, the Cache Manager continues to distribute the cached data until the time since verification equals or exceeds the configurable time period. The next time data is requested, the Cache Manager retrieves the newer version to its cache before providing it to the application program.

The **dfsd** process initializes the Cache Manager on a client machine. It can be used to alter aspects of the Cache Manager's cache, such as its location and size. It also starts several background daemons, which help the Cache Manager manage the data stored in its cache.

The **dfsbind** process, in addition to its role on File Server machines, is used by the Cache Managers on client machines to help with the resolution of DCE pathnames. It also obtains authentication information about users that Cache Managers require for RPC bindings to File Server machines. (See "Client Machine Processes and Files" on page 38 for more information about the Cache Manager and the **dfsd** and **dfsbind** processes.)

Exporting Data from a Client Machine (Private File Server Machine)

The primary function of a client machine is to communicate with File Server machines to access files for application programs. However, a client machine can also be configured as a Private File Server machine to export data from its local disk for use in the global namespace. To export data as a Private File Server machine, a client machine must meet the following additional requirements:

- Have an RPC binding in CDS
- Have a DFS server principal and associated account in the Registry Database
- Have a server entry in the FLDB
- Run the Fileset Server (**ftserver** process)
- Run the File Exporter, which is initialized with the **fxd** process
- Run the **upclient** process to retrieve binary files from the proper Binary Distribution machine
- Run the BOS Server (**bosserver** process)
- Optionally, run the Replication Server (**repserver** process)

Although meeting these requirements qualifies a client machine as a File Server machine, that is not the machine's primary role. The machine's local disk is not to

be used for data storage for an entire cell or domain. A client machine meets the previous requirements solely to allow users who administer the machine to make data on its local disk available in the global namespace. (See “File Server Machines” on page 30 for more information about these additional processes.)

To prohibit other users from creating filesets on the client machine, the users who administer the machine should be the only ones listed in the **admin.ft** list and the **fxd** administrative group for the machine. They should also be listed in the group that is given ownership of the server entry for the machine in the FLDB. These local privileges do not grant the owners of the workstation administrative privilege beyond the local machine. However, the owners have all of the privileges required to administer the filesets on their machine and the entries for those filesets in the FLDB. These privileges, and the ability to set the ACLs for any data that is exported from the workstation, allow the owners to prevent other users from storing data on the machine.

Because a client machine that exports data must run DFS server processes (such as **bosserver**, **ftserver**, and **fxd**), it must also run the **upclient** process to retrieve current versions of binary files for the processes from the Binary Distribution machine for its CPU/OS type in the cell. It must therefore be included in the **admin.up** list of the Binary Distribution machine of its CPU/OS type. Beyond that, neither the machine nor its owners need to be included in the administrative lists used by the other machines in their cell or administrative domain.

Summary of DFS Machine Roles

The following table summarizes the DFS machine roles described in the previous sections. For each machine role, the table provides a brief description of its purpose and lists the DFS processes that a machine filling the role must run. The table also provides suggestions for how to configure machines of a specific type and other roles a machine of each type can assume. A machine that is assuming any of the roles listed in the table must be configured as a DCE client machine. A machine assuming a role as a DFS server must have both an RPC binding in CDS for its pathname and a DFS server principal in the Registry Database.

Recall that any server machine can be configured to perform any of the other server machine roles. Also, a server machine can be configured as a client machine, and vice versa. To fill an additional role, a machine must run the processes listed for that role in the third column of the table. (See “Overview of DFS Machine Roles” on page 28 for expanded descriptions of the machine roles.)

Note: The following table uses the numbers **1** and **2** to differentiate the **upserver** and **upclient** processes running on the machines. The notations **upserver**[1] and **upclient**[1] denote the Update Server that distributes common configuration files from a System Control machine. The notations **upserver**[2] and **upclient**[2] denote the Update Server that distributes binary files from a Binary Distribution machine.

Table 1. Summary of DFS Machine Roles

Machine Role	Purpose	Processes	Suggestions
System Control machine	To distribute common configuration files for a domain	`bosserver` `upserver`[1] `upclient`[2]	Use a Binary Distribution machine as the System Control machine for a domain.

Table 1. Summary of DFS Machine Roles (continued)

Machine Role	Purpose	Processes	Suggestions
Binary Distribution machine	To distribute system binary files for its CPU/OS type	`bosserver` `upserver`[2] `upclient`[1]	Use the System Control machine for a domain as a Binary Distribution machine.
File Server machine	To export and store DCE LFS and non-LFS data	`bosserver` `ftserver` `fxd` `dfsbind` `repserver` `upclient`[1] `upclient`[2]	A File Server machine must also have a server entry in the FLDB. In a large cell, dedicate one File Server machine to housing read-only replicas.
Fileset Database machine	To store the Fileset Location Database (FLDB)	`bosserver` `flserver` `upclient`[1] `upclient`[2]	Configure three Fileset Database machines. Configure Fileset Database machines as Backup Database machines.
DFS client machine	To serve as a single-user or multiuser workstation; to access files for application programs	`dfsd` `dfsbind`	

DFS Server and Client Configuration Issues

The following subsections describe some general issues to consider before configuring DFS server and client machines. They also provide additional information about the files that must reside on server and client machines and a few of the processes only briefly described in earlier sections of this chapter. They also serve as an introduction to some issues to be considered before configuring a domain.

Server Machine Processes and Files

As mentioned previously, you should combine machine roles for the machines in your cell and domains. For example, you may want to set up a database server machine to house both the FLDB and the Backup Database. A machine that houses these databases needs to be stored in a secure location so that unauthorized users cannot access and possibly damage fileset data or the databases.

In any cell, there is only one version of the FLDB and one version of the Backup Database, even though these databases can be replicated at other sites. The initial copies of these databases are created when the Fileset Location and Backup Servers are first started in the cell. They are replicated to other machines automatically as additional instances of their respective server processes are started on those machines. When configuring a new domain in an existing cell, do not attempt to create a new FLDB or Backup Database for the domain; configure additional instances of the existing database as necessary.

Several directories contain files related to DFS server processes. The directories in the following list store files on a server machine's local disk. Files stored on the local disk are generally required for DFS to start without accessing the global namespace.

- The *dcelocal*/**bin** directory contains DFS binaries that are appropriate for the machine's CPU/OS type. The binary files are for server processes, command suites, and other processes and programs.
- The *dcelocal*/**var/dfs** directory houses administrative lists for server processes; for example, **admin.bos** and **admin.ft**. It also contains configuration files that are used by the BOS Server and the **dfsexport** command. If the machine is running the Fileset Location Server, this directory also contains the FLDB.
- The *dcelocal*/**var/dfs/adm** directory stores log files generated by server processes. These files detail events that occur during the operation of server processes. Server processes do not use these log files to reconstruct failed operations because only completed events are recorded in them. However, because the information in the files is in human-readable format, examination of these files is the first step in the troubleshooting procedure. They can help you evaluate process failures and related problems.

  The *dcelocal*/**var/dfs/adm** directory also contains the core image file that is generated if a process being monitored by the BOS Server crashes. The BOS Server adds an extension to the standard **core** name to indicate which process generated the file; for example, **core.flserver**. However, if two processes abort at exactly the same time, the BOS Server may not be able to assign the correct extension to the core file.

In addition, the *dceshared*/**bin** directory also stores all of the binary files that are housed in the *dcelocal*/**bin** directory. Current versions of the files are always available from *dceshared*/**bin** for installation on the local disk of a server machine. The directory also contains the binary files for a number of programs that are not integral to starting DFS, such as the **scout** program and a number of programs related to the DFS Backup System.

Client Machine Processes and Files

Client machines run the **dfsd** process, which initializes the Cache Manager, and the **dfsbind** process. You can save disk space on a client machine by storing commonly used files in the DFS filespace. You can then create symbolic links on the local disk that refer to the files in the filespace.

When the Cache Manager retrieves a requested file, it caches the data before passing it on to an application program. It does not cache the entire file; it instead caches ″chunks,″ or pieces, of data. By default, each chunk of cached data contains 64 kilobytes of data in a disk cache or 8 kilobytes of data in a memory cache.

The **dfsd** process initializes the Cache Manager on a client machine by transferring configuration information into kernel memory. It also mounts the root of the global namespace (**/...**). You can use the options available with the command line for the **dfsd** process to alter the definitions for the type of cache to be used (disk or memory), total cache size, cache chunk size, the local disk directory to be used for caching, and other configuration information.

In addition, the **dfsd** process starts several background daemons. These daemons include one or more maintenance daemons that perform routine maintenance tasks such as garbage collection, background daemons that improve performance by

performing delayed writing of updated data, token daemons that respond to token revocation requests from File Exporters, and (on the AIX operating system) I/O daemons that move data between disk and memory.

The **dfsbind** process resolves CDS pathnames and returns information about Fileset Database machines to the Cache Manager. The information allows the Cache Manager to contact the FL Server on an appropriate Fileset Database machine in the cell to determine the locations of filesets that house data requested by users.

The **dfsbind** process also returns user authentication information from the Security Server to the kernel RPC Runtime of the client machine. Authentication information must be included in RPC bindings that request data from a File Server machine for a user. The Cache Manager uses the RPC bindings to access data for the user from the File Server machine.

(See the Reference part of this guide and reference for complete information about the **dfsd** and **dfsbind** commands that start the respective processes and the options available with the commands.)

Two types of files must reside on the local disk of a client machine: boot sequence files needed during reboot, and files that are useful during File Server machine outages.

During a reboot, DFS is inaccessible until the **dfsd** process reinitializes the Cache Manager; the **dfsbind** process must be running before the **dfsd** process can be run. Any files needed during reboot and before the **dfsd** process starts must reside on the local disk. Following is a list of recommended DFS files to store on a local disk.

- The *dceloca*l**/bin/dfsbind** command is the start-up command for the **dfsbind** process.
- The *dceloca*l**/bin/dfsd** command is the start-up command for the Cache Manager.
- The *dceloca*l**/etc/CacheInfo** file is a file that specifies aspects of Cache Manager configuration.
- The *dceloca*l**/var/adm/dfs/cache** directory is a directory that contains cache-related files, such as **V** *n* files and the **CacheItems** file, generated and used by the Cache Manager.

You may also want to store diagnostic and recovery files on a local disk. Certain commands in the **bos** and **cm** command suites can help users diagnose problems caused by a File Server outage. It is useful to have local disk copies of the binary files for the **bos** and **cm** suites because the File Server outage that requires their use can also make them inaccessible. In addition, you may want to keep the binaries for a text editor, such as **ed** or **vi**, on the local disk for use during outages.

Additionally, if you wish to modify the default Cache Manager preferences for accessing File Servers and FLDB machines, you can add **cm setpreference** commands to the machine's initialization file. Doing so ensures that such preferences are loaded each time the machine is initialized. For more information about Cache Manager preferences for File Servers and FLDB machines, see the following section.

Note: If you are using DFS, you must use **dce_login** after the DFS client is configured and running on your machine in order to have authenticated access to DFS objects. Any credentials acquired before DFS is running are not recognized by DFS.

Multihomed Server Configuration Issues

Multihomed server capabilities allow administrators to specify up to four interfaces (either hostnames or IP addresses) in the FLDB for each File Server and FLDB machine. Servers can have more than four network connections; however, the FLDB can accept only four entries per server. This capability, coupled with server preference lists maintained by the individual Cache Managers, allows you to configure DFS to work optimally within your network.

For example, a single File Server can have up to four IP addresses (specified for use by DFS), and the various clients that use that server can have their Cache Manager preference lists configured so that the preferred access to that server is through the most efficient possible network connection. Should a single connection to a File Server become unavailable, the various clients that previously used that connection would consult their Cache Manager's preference lists and reroute their requests to another address for a File Server containing the required fileset. This behavior lets you configure DFS for the most efficient use of the network while providing additional fail-over capabilities for the file system.

How Multihomed Servers and Preferences Work Together

Each Cache Manager maintains a list of File Server and Fileset Location (FL) Server preferences. Each entry in that list contains both the address of a server and a ranking. The ranking value determines the order in which these servers are accessed, or their ″preference.″ The FLDB can contain up to four addresses for each server machine; therefore, the preference list can also contain up to four entries for each server (each with its own address and preference rank).

In operation, when a Cache Manager requires a particular fileset, it first consults its list of FL Servers and attempts to contact an FL Server at the address with the lowest ranking in the preference list. The FL Server provides the addresses of the various File Servers that contain that fileset. (The fileset location information is then cached by the Cache Manager and is updated periodically.) If the fileset is replicated, multiple File Servers may contain that fileset. The Cache Manager again consults its preference list and contacts a suitable File Server at the address with the lowest ranking value. Should the Cache Manager not be able to contact a server during this process, it simply checks its preference list and attempts to contact a suitable server at the next most preferred IP address.

The preference list is created automatically each time a Cache Manager is initialized. It consists of the IP addresses of FL Servers and File Servers and an automatically assigned preference value for each. New entries are added to the preference list as necessary when filesets are first required. By default, the Cache Manager assigns preferences that make sensible choices based on the location of servers. The default values make the Cache Manager try to connect to servers in the following order:

1. The same machine as the client (default rank of 5000).
2. The same subnetwork as the client (default rank of 20,000).
3. The same network as the client (default rank of 30,000).
4. Different networks (default rank of 40,000).

Cache Manager preferences are explained in detail in “Chapter 8. Configuring the Cache Manager” on page 217.

For example, a server on the same machine as the Cache Manager receives a rank of 5000, while a server on the same subnetwork receives a rank of 20,000. The entry with the lowest ranking value has the highest preference. Thus, a server with a preference value of 5000 will be chosen before a server with a rank of 20,000.

You can change Cache Manager preferences by using the **cm setpreferences** command. Additionally, you can create a file specifying server preferences that is read each time a Cache Manager is initialized, thus providing a method for overriding the default server preference values. You can also load preference entries from standard input or any combination of all three sources. This procedure is also explained in “Chapter 9. Configuring the Backup System” on page 241.

Should two servers be assigned the same preference value, such as two File Servers on the same subnetwork both receiving a default value of 20,000, the server with the lowest round-trip value is chosen. Each server is assigned a random round-trip value when the Cache Manager is initialized. The assigned round-trip value is always higher than the upper bound for stored actual round-trip values. This ensures that an actual round-trip value is always chosen over assigned values.

By judiciously providing multiple addresses for FL Servers and File Servers and properly configuring the Cache Manager preference lists, you can configure DFS to make the most efficient use of servers within the network. For example, you may want to provide a connection from commonly used File Servers and FL Servers to the same subnetworks that are shared by the majority of the DFS clients. This connection reduces cross-router and gateway traffic through the network. As a backup, you can provide higher-ranking preference entries for server connections to other areas of the network. This configuration provides continued access to the servers should a particular network connection become unavailable.

The following simplified scenario illustrates how multihomed servers can be configured to make the most efficient use of the local network. In this example, a read-only fileset is replicated on two File Servers. The File Servers have connections to both subnetworks within the network, and these connections are the preferred connections used by DFS clients on each respective subnetwork. When a DFS client must fetch data from the read-only fileset, it first consults the list of suitable Files Servers. The Cache Manager then consults its list of preferences and chooses the connection to a suitable File Server that has the lowest rank. Because both File Server connections on the local subnetwork have the same rank, the connection with the lowest round-trip value is chosen, as shown in Figure 1 on page 42 .

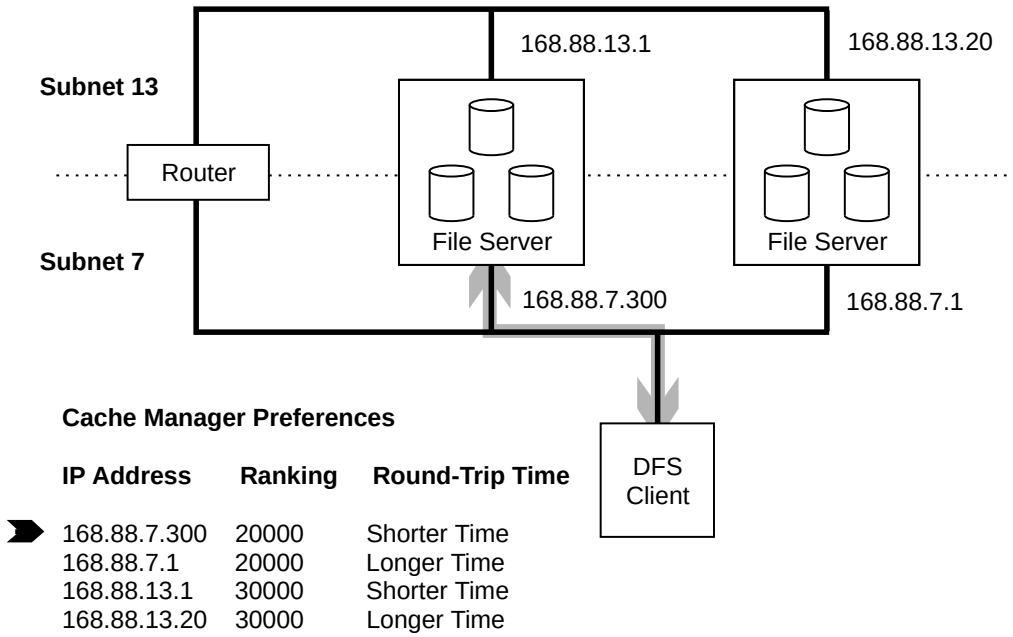


Figure 1. Cache Manager Contacting File Server Address With Lowest Rank

Should the Cache Manager lose contact with the preferred File Server connection (either through a network or server problem), the Cache Manager again consults its preference list and attempts to contact a suitable File Server at the address with the next lowest rank, as shown in Figure 2. In this figure, when the Cache Manager can no longer contact a File Server at a given connection, it attempts to connect to the File Server address with the next-lowest preference value.

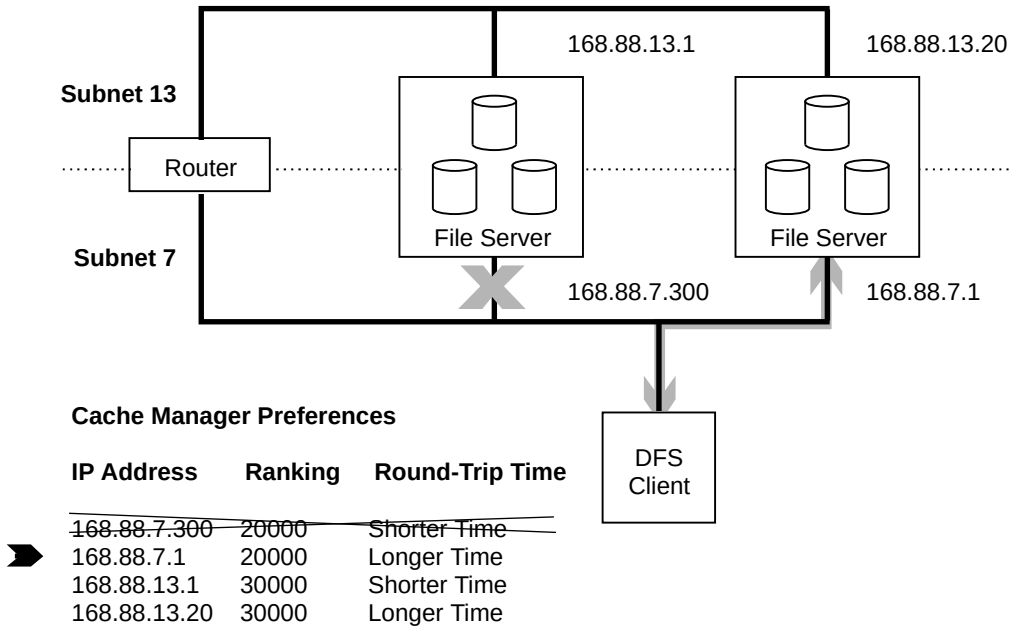


Figure 2. Cache Manager Connecting to File Server Address With Next Lowest Rank

If the Cache Manager again loses contact with a File Server through its current connection, it once more consults the preference list for the address of a suitable File Server with the next lowest value. In this case, the Cache Manager must now establish a connection to another subnetwork. There are two possible connections to suitable File Servers in that subnetwork, both having the same rank. The Cache Manager, therefore, chooses the connection with the lowest round-trip time value, as shown in Figure 3. In this figure, should the Cache Manager again lose its connection, it checks the preference list for a connection to a suitable File Server with the next lowest ranking.

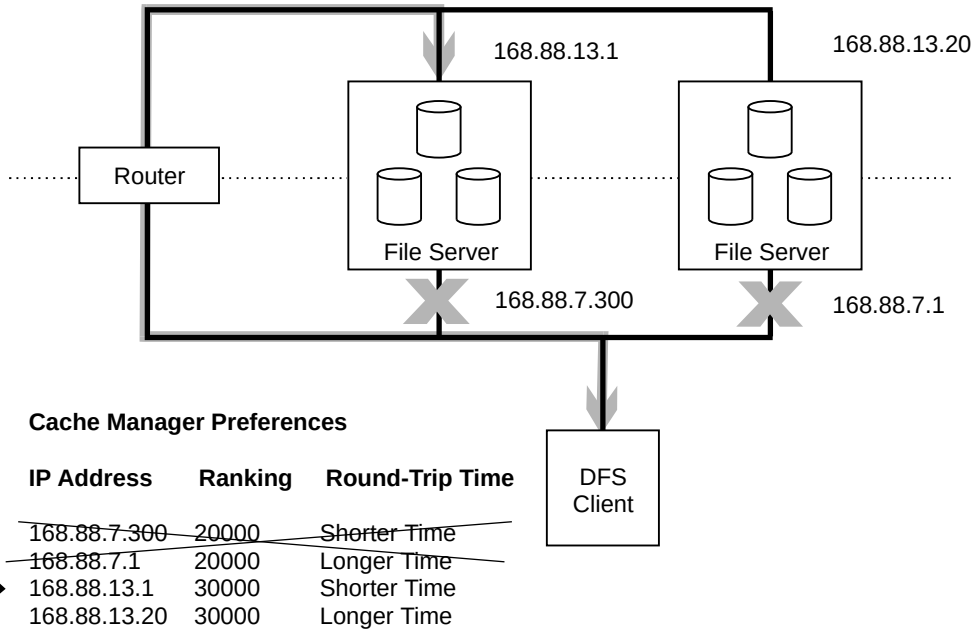


Figure 3. Cache Manager Again Losing Connection and Contacting File Server Address in Another Subnet

The entire process of changing connections as required is carried out automatically, without the DFS client users being aware that it has occurred.

Tasks to Administer Multihomed Servers

The following are tasks to configure a multihomed File Server environment:

- You must add each host name or IP address for each File Server to the Fileset Location Database (FLDB). The initial entry, along with one address for that entry, is created using the **fts crserverentry** command. Additional addresses can be added or deleted from the entry by using the **fts edserverentry** command. See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for more information.
- Optionally, you can modify the preferences for each client’s Cache Manager to take advantage of the most efficient connections to the File Servers. However, you should modify the preferences only when there are compelling reasons do so. The Cache Manager’s default preferences are generally the most efficient for any given network configuration.

  Client preference lists are transient in that they are reestablished at their default values each time the Cache Manager is initialized. However, a list of preferences can be loaded into the Cache Manager at initialization through a preferences file.

"Chapter 8. Configuring the Cache Manager" on page 217 explains how to both create such a file and ensure that it is loaded each time the Cache Manager is initialized.

IP Layer Override of Preferences

While the FLDB can only contain up to four addresses for a given File Server or FL Server, such servers can have more than four connections to the network. In such instances, the DCE RPC mechanism can allow the IP layer to choose a source address for a server response that is different, and presumably more efficient, than the specified destination of the corresponding request. In this case, the chosen server address is likely to be a function of the client address to which the response is being set; however, the exact algorithm for choosing the address will differ for each operating system vendor. Such a routing decision is observed by the Cache Manager as a change in the server-binding's address.

Should the IP layer select a different server address, this connection becomes the connection used by the Cache Manager, regardless of its preference rank or whether or not it is one of the addresses listed in the FLDB for a given server. This scenario is shown in Figure 4.

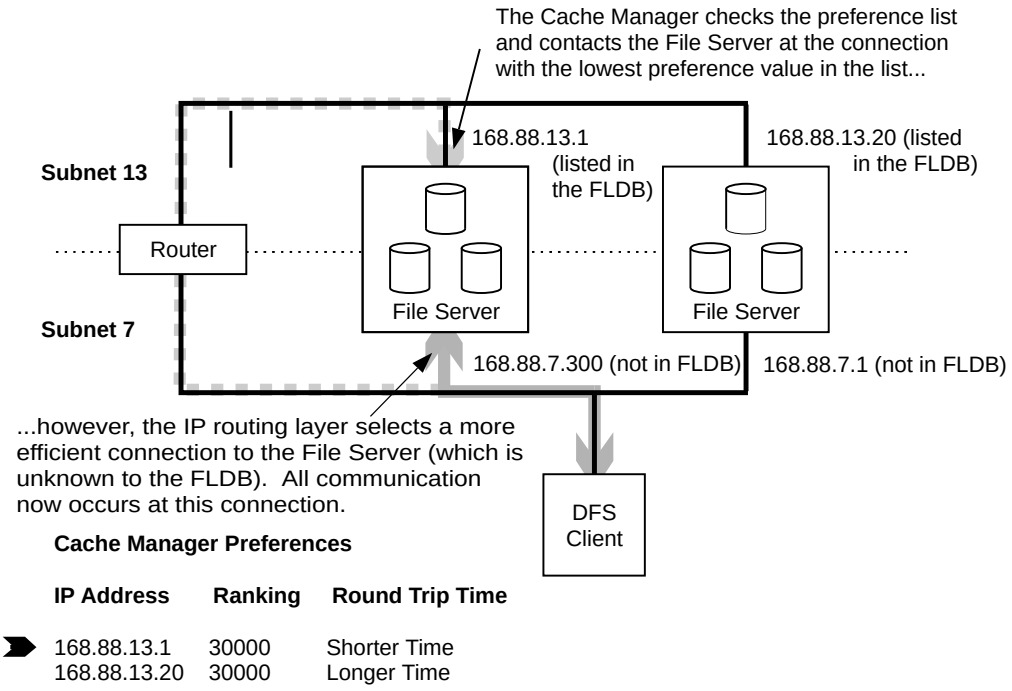


Figure 4. An Example of the IP Layer Overriding the Cache Manager's Preference

Should the IP layer select a different connection and override the preference choice, the **cm getpreferences** command returns the address of the currently used connections (the connection selected by the IP layer) as the entry in the preference list, even though it may not be listed in the FLDB.

Creating Additional Default Entries in the Routing Table

The preference list provides each Cache Manager with a list of known connections to various File Servers and FL Servers. This list allows the Cache Manager to

select alternative connections to communicate with the appropriate servers should a network or server fault make a particular connection unavailable. Similarly, each File Server or FL Server that has multiple connections to the network should have multiple default entries in its routing table that define the various routers available to that server. Thus, if a network fault makes a particular router unavailable, that server has additional router choices that would allow it to reply to Cache Manager requests. Refer to your operating system documentation for information concerning adding default entries to a server's routing table.

Setting Up Filesets

DCE LFS filesets are created with the **fts create** command. Non-LFS filesets are created in the local operating system and registered in DFS with the **fts crfldbentry** command. Mount points to the global namespace for both DCE LFS and non-LFS filesets are created with the **fts crmount** command.

The following subsections discuss setting up a cell's root fileset, binary and configuration filesets, and user filesets. Information about fileset replication and the **@sys** and **@host** variables, which simplify cell administration, is also provided. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for complete information about creating and mounting filesets; see the Reference part of this guide and reference for complete information about **fts** and other DFS commands.)

Note: DCE 2.2 for AIX provides additional commands to perform initial DFS configuration tasks and some DFS administrative tasks. For specific information about the **config.dfs**, **mkbutc.dfs**, **mkfilesys.dfs**, **rmfilesys.dfs**, **rmbutc.dfs**, and **unconfig.dfs** commands, see the Reference part of this guide and reference. For more information about using the AIX SMIT utility to configure and manage DFS, see *IBM DCE for AIX, Version 2.2: Quick Beginnings*.

Setting Up the Root Fileset

The main read/write fileset, **root.dfs**, is required in every cell's file system. It is the first fileset created in a cell during DFS configuration. It is the implied fileset for the root of a cell's DFS filespace (***/.../*** *cellname****/fs***, by default). It can be a DCE LFS fileset or it can be a non-LFS fileset. However, it must be a DCE LFS fileset if functionality such as replication is to be available in the cell.

To create **root.dfs** as a DCE LFS fileset, issue the **fts create** command to create the fileset on a specified server machine and exported DCE LFS aggregate. For example:

$ fts create root.dfs -server *machine* **–aggregate** *name*

Once the root fileset is created, start the **dfsd** process if it is not already running. The **dfsd** process mounts the root of the global namespace (***/...***) automatically. Once the global namespace is mounted, the **root.dfs** fileset resides at the top level of the cell's DFS filespace.

You must enter the **fts crmount** command with the **-rw** option to create an explicit read/write mount point for the fileset below the top level of the cell's DFS filespace. For example:

$ fts crmount /:/.rw root.dfs -rw

Once these steps are complete, you can replicate **root.dfs**. Replication is then available for DCE LFS filesets created in the cell. It is important that you follow these instructions if you plan to replicate filesets in your cell. Due to the nature of mount points, if you replicate **root.dfs** before creating its read/write mount point, you effectively make it impossible to access the read/write version of **root.dfs**.

(See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for more information about creating and mounting filesets, using mount points, and creating and exporting aggregates.)

Note: By default, the junction to the DFS filespace is defined at ***/.../*** *cellname***/fs**. However, the name of the junction is not considered to be well known and can be changed during installation and configuration of DCE. (See your vendor's installation and configuration documentation for more information.) The examples in this part of the guide use the default, **fs**, as the junction of the DFS filespace.

Choosing Fileset Names

Each directory in ***/.../*** *cellname***/fs** usually corresponds to a separate, mounted fileset (mounted filesets can also occur anywhere in the file tree). Subdirectories of ***/.../*** *cellname***/fs/** *directory_name* can be either standard directories or mount points to separate filesets. For simplified administration, group the directories and their contents into small, easily managed filesets.

Each fileset has a name unique to the cell in which it resides. Fileset names are stored in the FLDB. A fileset's name is not the same as the name of its mount point, although you can assign the same name to a fileset and its mount point.

There is a 111-character limit on the length of fileset names. However, because a 9-character **.readonly** extension is added when you replicate a fileset, you need to specify fileset names that contain no more than 102 characters. When creating filesets, do not add the **.readonly** and **.backup** extensions yourself; DFS automatically adds the appropriate extension when it creates a read-only or backup fileset. (DFS reserves the **.readonly** and **.backup** extensions for use with read-only and backup filesets. You cannot create a fileset whose name ends with either of these extensions.)

You can give filesets any names that you feel are appropriate. For simplified administration, however, a fileset's name needs to

- Reflect the fileset's contents
- Reflect the name of the fileset's mount point
- Be consistent with other filesets that contain similar types of data so that you can easily manipulate groups of filesets when using the DFS Backup System

You may find it helpful to use a common prefix for related filesets. The following list summarizes this type of naming scheme:

- Use the **common.** *type* prefix for common filesets. For example, use **common.etc** for common configuration files (mounted at **/.../** *cellname***/fs/common/etc**), and **common.forms** for common forms (mounted at **/.../** *cellname***/fs/common/forms**).
- Use the **src.** *type* prefix for source filesets. For example, use **src.dfs** for DFS source files (mounted at **/.../** *cellname***/fs/src/dfs**).
- Use the **user.** *username* prefix for all user filesets. For example, use **user.terry** for user **terry**'s fileset (mounted at **/.../** *cellname***/fs/usr/terry**).
- Use the **public.** *username* prefix for each user's public fileset. For example, use **public.terry** for **terry**'s public fileset, which contains information the user wants to make available to everyone. The **public.terry** fileset is mounted at **/.../** *cellname***/fs/public/terry**.
- Use the *sys_type***.** *distribution_dir* prefix for operating-system-specific filesets. For example, use **pmax_osf1.bin** for OSF/1 binary files (mounted at **/.../** *cellname***/fs/pmax_osf1/bin**), and **pmax_osf1.lib** for OSF/1 library files (mounted at **/.../** *cellname***/fs/pmax_osf1/lib**). In DFS, symbolic links are often created from the **/bin** and **/lib** directories (or their equivalents) on the local disk of a workstation to these DFS mount points.

(See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for more information on additional rules for naming filesets.)

Setting Up Binary and Configuration Filesets

You may find it convenient to store DCE binaries, system binaries, and configuration files (for example, those commonly found in directories such as **/bin** and **/etc** or their equivalents) in the DFS filespace, instead of on the local disk of each machine. Because binary files are operating-system specific, you may want to create a different fileset for each system type (for example, **pmax_osf1** or **rs_aix41**) and distribution directory (for example, **/etc** and **/bin**) and store the filesets on a DFS File Server machine. You can then create symbolic links from the local disk to the fileset.

Note that DFS simplifies the creation of such links by providing the **@sys** variable, which is set on a per-Cache Manager basis. When the Cache Manager encounters the **@sys** variable in a pathname, it substitutes its system name for the variable. (See "Using the @sys and @host Variables" on page 50 for a more detailed description of the **@sys** variable.)

For example, while it is a good practice to store the binary files for a single text editor on the local machine, the binaries for other text editors do not need to be stored on each machine. A system administrator can create filesets that store text editor binaries for each system type. The administrator can then construct a symbolic link from the local disk of each machine to the appropriate fileset in DFS. For instance, system administrators in the **abc.com** cell running the Solaris 2.3 (SunOS 5.3) and AIX 4.1 operating systems, can configure part of their file tree as shown in the following table.

Table 2. Examples of Fileset Names and Mount Points for Binary Files

Fileset Name	Mount Point
sparc_sunos53	/.../abc.com/fs/sparc_sunos53
sparc_sunos53.bin	/.../abc.com/fs/sparc_sunos53/bin
sparc_sunos53.etc	/.../abc.com/fs/sparc_sunos53/etc

Table 2. Examples of Fileset Names and Mount Points for Binary Files (continued)

Fileset Name	Mount Point
rs_aix41	/.../abc.com/fs/rs_aix41
rs_aix41.bin	/.../abc.com/fs/rs_aix41/bin
rs_aix41.etc	/.../abc.com/fs/rs_aix41/etc

Storing common files in a central location eliminates the need to store copies on every client's local disk, which in turn saves local disk space. Replication further enhances the availability of common files. (Some binaries, however, must remain on the local disk of every machine.)

Setting Up User Filesets

Each user has a unique DCE account. You may also want to create a single, separate fileset for each user and mount the fileset at **/.../** *cellname***/fs/usr/** *username*, where *username* is the name of the user who owns the fileset. For example, assign the name **user.terry** to the fileset for the user named **terry**. When you mount the fileset at **/.../abc.com/fs/usr/terry**, the root directory of the fileset (the user's home directory) is named **/.../abc.com/fs/usr/terry**. The user's home directory contains all of the files, subdirectories, and mount points in the fileset named **user.terry**.

As with any other fileset, you may want to create additional filesets based on logical file groupings and mount them below **/.../** *cellname***/fs/usr/** *username* if the user's fileset becomes too large. For example, if **terry** has 5000 kilobytes of data in the **project1** subdirectory and 3000 kilobytes of data in the **project2** subdirectory, you may want to create two smaller filesets organized below **/.../abc.com/fs/usr/terry**. The following table lists the organization and names of the filesets in this example.

Table 3. Examples of Fileset Names and Mount Points for User Data

Fileset Name	Mount Point
user.terry	/.../abc.com/fs/usr/terry
user.terry.project1	/.../abc.com/fs/usr/terry/project1
user.terry.project2	/.../abc.com/fs/usr/terry/project2

Moving Data from Non-LFS Directories to DCE LFS Directories

The guidelines in "Setting Up User Filesets" assume that the user does not have an existing home directory in the file system. A user who has data in an existing home directory in a non-LFS fileset mounted in the global namespace can continue to use that fileset. However, if you choose to create and mount a DCE LFS fileset for the user, you must be careful: DFS does not allow you to mount a fileset at an existing directory. You must move the user's data from the existing home directory in the DCE namespace to a temporary directory. You must then remove the existing home directory before creating and mounting the user's DCE LFS fileset. You can then move the user's data to the new fileset.

For example, suppose the user named **terry** in the previous example has an existing home directory in a non-LFS fileset mounted at **/.../abc.com/fs/usr/terry**. In this case, move the user's data from **/.../abc.com/fs/usr/terry** to a temporary location (such as a subdirectory of **/tmp** on the local disk), and remove the **/.../abc.com/fs/usr/terry** directory and its contents. Then create and mount the

user's DCE LFS fileset as described in "Setting Up User Filesets" on page 48. Finally, move the user's data from the temporary location into the new DCE LFS fileset mounted at **/.../abc.com/fs/usr/terry**. When these steps are complete, the user can access the data as before.

Replicating DCE LFS Filesets

You replicate DCE LFS filesets by placing read-only copies of them on one or more File Server machines in a cell. If a machine that houses a read-only copy of the fileset becomes unavailable, the information is usually still available from a copy of the fileset on another machine. However, for a fileset that uses Release Replication, if the read-only fileset that resides at the same site as the read/write fileset becomes unavailable, all other read-only versions of that fileset become unavailable after a configurable amount of time. Similarly, for a fileset that uses Scheduled Replication, if the read-write fileset becomes unavailable, all read-only versions of the fileset become unavailable after a configurable amount of time. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for detailed information on the availability of read-only filesets.)

Replicate only those DCE LFS filesets that meet the following criteria:

- The files in the fileset are read much more often than they are modified.
- The files in the fileset are used heavily (for example, binary files for text editors or other heavily used application programs). Replicating the fileset lets you distribute the load for the files that it contains across several machines.
- The files in the fileset must remain available. By replicating the fileset on multiple File Server machines, even if one of the machines that houses a replica of the fileset becomes unavailable, replicas are usually still available from other machines.
- The fileset is mounted at a high level in the cell's file tree; for example, **root.dfs** and its subdirectories.

If your cell is large, you may want to use a small set of File Server machines to store just read-only filesets. These machines can then distribute frequently used data, reducing the load on other machines. Keep in mind that each replica not stored on the same aggregate as its read/write source fileset uses as much disk space as its source fileset. A read-only fileset created on the same aggregate as its source fileset is created as a clone of its source and so requires potentially much less space than a full read-only replica created on a different aggregate.

Each Cache Manager maintains preferences in the form of numerical ranks that bias its selection of File Server machines for read-only fileset access. When accessing a read-only fileset, the Cache Manager consults its collection of preferences and attempts to access the read-only fileset from the File Server machine that has the lowest recorded rank. If the Cache Manager cannot access the fileset from that machine, it tries to access the fileset from the machine that has the next-lowest rank. It continues in this manner until it either succeeds in accessing the fileset or determines that all of the machines that house the fileset are unavailable.

By default, the Cache Manager assigns preferences to File Server machines based on IP addresses. You can set or change the Cache Manager's preferences to suit your needs. (See "Chapter 8. Configuring the Cache Manager" on page 217 for more information about the Cache Manager and File Server machine preferences.)

Using the @sys and @host Variables

DFS simplifies the administration of operating system-specific or host-specific files by providing the **@sys** and **@host** variables. When the Cache Manager encounters **@sys** or **@host** in a pathname, it replaces the variable with either the system name (defined with the **cm sysname** command) or the hostname (defined with the local operating system's **hostname** command or its equivalent).

The **@sys** and **@host** variables are especially useful when constructing symbolic links from the local disk to the DFS filespace. You create identical links on all machines, yet each machine accesses the files that are appropriate to its system type or hostname. Use the **@sys** variable to access files that are organized on a per-system type basis; use **@host** to access files that are organized on a per-machine basis. The following subsections provide examples of these variables.

The @sys Variable

The **@sys** variable is expanded to the name of a CPU/OS type. The **cm sysname** command sets and displays the current value of the **@sys** variable. The following examples show how the Cache Manager interprets the same pathname differently, depending on the value of **@sys**.

On a machine running Solaris 2.3 (SunOS 5.3):

```
$ cm sysname
Current sysname is 'sparc_sunos53'

$ cd /.../abc.com/fs/@sys

$ pwd
/.../abc.com/fs/sparc_sunos53
```

On a machine running AIX 4.1:

```
$ cm sysname
Current sysname is 'rs_aix41'

$ cd /.../abc.com/fs/@sys

$ pwd
/.../abc.com/fs/rs_aix41
```

The **@sys** variable is commonly used in symbolic links from a DFS client machine to a fileset in the DFS filespace. A single copy of a binary file for each system type is stored on a single File Server machine in DFS instead of on the local disk of each client machine. Links are then created from client machines to the central copy of the binary file, eliminating the need to store the same binary file on each client machine. Accessing binary files this way saves disk space on client machines and ensures that users on all client machines are using the same version of the binary file. It also eases system administration by allowing administrators to update central copies of binary files, rather than requiring them to update the copies stored on each client machine.

A link that includes the **@sys** variable can be created on each client machine. The Cache Manager on each machine interprets the **@sys** variable, so each machine

accesses the binary file for its system type from the global namespace. Symbolic links that include the **@sys** variable are commonly used to access binary files for programs such as **make** and **emacs**.

The following examples create a symbolic link used to access the proper binary files for programs traditionally stored in **/usr/local**. In the examples, the Cache Managers on two machines interpret the link differently, depending on their respective values of **@sys**.

On a machine running Solaris 2.3:

$ ln -s /.../abc.com/fs/@sys/usr/local /usr/local $ ls −l /usr/local

```
lrwxrwxrwx 1 root 32 Aug 1 06:44 /usr/local ->
/.../abc.com/fs/@sys/usr/local
```

$ cd /usr/local

$ pwd

```
/.../abc.com/fs/sparc_sunos53/usr/local
```

On a machine running AIX 4.1:

$ ln -s /.../abc.com/fs/@sys/usr/local /usr/local $ ls −l /usr/local

```
lrwxrwxrwx 1 root 32 Aug 1 06:44 /usr/local ->
/.../abc.com/fs/@sys/usr/local
```

$ cd /usr/local

$ pwd

```
/.../abc.com/fs/rs_aix41/usr/local
```

When creating links on server machines, do not use links to access binary files for DFS server processes. These files must reside on the local disk of each server machine to avoid bootstrapping problems.

(See the Reference part of this guide and reference for more information about the **cm sysname** command.)

The @host Variable

The **@host** variable is expanded to the value defined by the **hostname** command (or its equivalent) of the local operating system. The **@host** variable is especially useful when configuring machines that must execute a machine-specific set of start-up routines.

For example, suppose two machines, **fs1.abc.com** and **fs2.abc.com**, use two different, machine-specific versions of an initialization file for an application that they start following a reboot. The name of the initialization file is **start**. The file **start** can be stored in DFS and accessed on a machine-specific basis via the **@host** variable. To access the proper copy of the file, both machines can have symbolic links from **/etc/rc/start** to **/.../abc.com/fs/etc/@host/rc/start**. On the first machine, the symbolic link resolves to the file named

/.../abc.com/fs/etc/fs1.abc.com/rc/start

On the second machine, the symbolic link resolves to the file named

/.../abc.com/fs/etc/fs2.abc.com/rc/start

Data Access Management in DFS

All access to data and metadata on a File Server machine is managed by the File Exporter. Clients contact the File Exporter to access data. The Cache Manager is the client of the File Exporter most visible to the user, as well as the one most frequently discussed in this guide, but other clients do exist. For example, the **fts** program can become a client of the File Exporter when a fileset is moved from one aggregate or machine to another, and the Replication Server is a frequent client of the File Exporter as it manages replicas of read/write filesets.

The File Exporter uses tokens to manage the distribution of data and metadata to clients. A client that wants to access or change data must first request and obtain the proper tokens for the data from the File Exporter on the machine on which the data resides. If the File Exporter can grant the client's request, it passes the tokens to the client; otherwise, it either queues the request for service later or rejects the request. A client that receives the requested tokens can then use them to access the data it wants from the File Exporter.

The following subsections provide more detailed information about tokens, their management by the File Exporter, and the token state recovery that occurs after a communications failure between a File Exporter and its clients.

Tokens

Tokens and their distribution and management by the File Exporter are completely transparent at the user level. The File Exporter uses tokens to

- Track the clients to which it has given data and the types of operations they are permitted to perform on the data.
- Ensure that multiple clients are not simultaneously accessing the same data in a conflicting manner.
- Guarantee that each client always has access to the most recent versions of read/write data. If data stored on a File Server machine changes while a client has a copy of it, the File Exporter on that machine notifies the client; the client then obtains the new version of the data the next time the data is needed.

Different operations require different types of tokens. DFS includes four general classes of tokens:

Open Tokens
: Allow a client to open an entire file or fileset to read from it, write to it, delete it, or prevent it from being deleted. For example, a client that wants to open a file for reading requests an open token for the file.

Status Tokens
: Allow a client to read or write file status information. For example, a client that wants to append data to a file, thus changing its size, needs a status token for the file.

Data Tokens
: Allow a client to read from or write to a range of bytes in a file. For example, a client that wants to modify the first 10 bytes of a file requests a data token for those bytes.

Lock Tokens
: Allow a client to read lock or write lock a range of bytes in a file. For

example, a client that must ensure that only one process is locking the first 10 bytes of a file requests a lock token for those bytes.

Each token class includes a number of token types; for example, the data class includes the read data and write data types. The different classes and types of tokens combine to allow for the different kinds of data access required by file system clients. Most operations require that a client possess multiple tokens for the data it wants to manipulate; for instance, appending text to a file requires open tokens to access the file, data tokens to modify the contents of the file, and status tokens to change the size of the file.

Some tokens can be granted to different clients simultaneously, while others cannot. Two tokens that can be granted simultaneously are said to be *compatible*; two tokens that cannot be granted at the same time are said to be *conflicting*. A token is always compatible with tokens from other classes, but it may conflict with other token types from within its class. In general, the token types associated with read operations are mutually compatible, while those associated with write operations conflict with other tokens.

Token Management

To determine whether it can grant a client's request for tokens, the File Exporter checks for outstanding tokens that conflict with those requested. If no other client has conflicting tokens, the File Exporter grants the requested tokens. If another client has conflicting tokens, the File Exporter takes the action associated with the first condition met from the following list:

- If the existing tokens can be revoked, the File Exporter revokes them and grants those requested. When its tokens are revoked, a client such as the Cache Manager flushes cached data for which the tokens applied, writing any modified data back to the File Server machine.
- If the existing tokens cannot be revoked, the File Exporter either places the request in a queue, to be serviced as soon as possible, or refuses to grant the requested tokens outright. The client dictates the File Exporter's response to this situation when it requests the tokens.

In general, if a client's existing tokens conflict with those requested by another client, the File Exporter attempts to revoke the existing tokens to grant the request. Many factors influence the File Exporter's ability to revoke a client's tokens. The File Exporter can usually revoke some types of tokens, but clients can refuse to relinquish other types of tokens in various situations. In addition, lifetimes that the File Exporter assigns to the tokens it grants and to the clients to which it grants them also affect its ability to revoke tokens, as follows:

Token Lifetime
: Specifies the length of time for which a token is valid. All tokens have a fixed token lifetime. Once its lifetime has elapsed, a token expires. The File Exporter needs to revoke only valid tokens. Because expired tokens are no longer valid, the File Exporter does not need to revoke them; it can simply grant new tokens as if the expired tokens did not exist. A client can contact the File Exporter to request that its tokens' lifetimes be extended before they expire.

Host Lifetime
: Indicates the length of time for which the File Exporter considers a client to be alive. Each client that has tokens from the File Exporter has a host lifetime within which it must contact the File Exporter to let it know that it is

still alive, thus renewing its host lifetime. The File Exporter needs the client's permission to revoke tokens that are held by the client as long as the client's host lifetime has not expired.

Host RPC Lifetime
Defines the length of time for which the File Exporter guarantees to attempt to make an RPC to a client before the File Exporter revokes its tokens. If the client responds to the RPC (thus renewing its host lifetime), the File Exporter cannot revoke the client's tokens without the client's permission. If the client fails to respond to the RPC but its host lifetime has not expired, the File Exporter cannot revoke the client's tokens; if the client fails to respond and its host lifetime has expired, the File Exporter can revoke any tokens the client holds without attempting to contact it further. The File Exporter can revoke the tokens of any client whose host RPC lifetime has expired without contacting the client; the client needs to either reclaim its tokens or request new ones as necessary.

Each File Exporter defines the lengths of its clients' host lifetimes and host RPC lifetimes, so a client can have different lifetimes for different File Exporters. For any File Exporter, however, a client's host RPC lifetime must be equal to or greater than its host lifetime. (By default, both lifetimes are only a few minutes in length.)

The following general rules govern the File Exporter's revocation of valid tokens held by a client:

- If the client's host lifetime has not expired, the File Exporter tries to contact the client; the File Exporter must have the client's permission to revoke its tokens.
- If the client's host lifetime has expired but its host RPC lifetime has not, the File Exporter tries to contact the client one time. If the client responds, the File Exporter cannot revoke the client's tokens without its permission; otherwise, the File Exporter can revoke any tokens the client holds without contacting it further.
- If the client's host RPC lifetime has expired, the File Exporter can revoke the client's tokens without contacting it.

Token State Recovery

Token state recovery refers to clients regaining their tokens following a communications failure between themselves and a File Exporter. The following problems can interrupt communications between a File Exporter and its clients:

- If a File Exporter is restarted (for example, after its File Server machine crashes), it loses all knowledge of the tokens it granted prior to the restart. For a brief period after it first returns to service, the File Exporter refuses all requests for new tokens from all clients, accepting requests only to reestablish tokens from those clients that held them before the File Exporter became unavailable. This is the first form of token state recovery.
- If a network failure prevents a client from contacting a File Exporter, the client may be unable to prevent its host lifetime from expiring. Once communications are restored, the client must either reclaim its tokens or, if necessary, request new ones. This is the second form of token state recovery.
- If a client is restarted, it loses all knowledge of the tokens it possessed prior to the restart; recovery of its tokens is not possible.

During the first form of token state recovery, the File Exporter attempts to preserve the state of its tokens across restarts by initially accepting requests only to reestablish existing tokens. While the File Exporter is unavailable, clients that have tokens from it continue to probe it at regular polling intervals until it returns to

service. When it is again available, the File Exporter enters token state recovery to give these clients the opportunity to recover their tokens without threat of conflicts with tokens that were granted to new clients.

Different File Exporters remain in token state recovery for different lengths of time after a restart. However, each File Exporter ensures that its recovery period lasts long enough to give all of its clients the opportunity to reestablish their tokens, basing the duration on the host lifetimes or polling intervals that it assigns, whichever are greater.

During the second form of token state recovery, the File Exporter does not provide the client with an opportunity to reestablish its tokens without fear of conflicting tokens. The client continues to poll the File Exporter until the network outage is resolved. However, if its host lifetime expires before it can contact the File Exporter, the client may be unable to recover tokens that it held prior to the network problem.

Values that the File Exporter uses to determine the host lifetimes, host RPC lifetimes, and polling intervals of its clients are specified with options of the **fxd** command. (See the Reference part of this guide and reference for complete information about the **fxd** command and its options.)

Data Communication Security in DFS

DFS includes administrative commands to establish and modify RPC authentication levels for communications between Cache Managers and File Servers. DFS provides commands for managing these RPC authentication levels, allowing you to set RPC authentication levels for each Cache Manager and RPC authentication bounds for each File Server. You can also set advisory RPC authentication bounds for each fileset.

Note: Higher authentication levels result in some degradation of performance (due to increased overhead).

Each Cache Manager maintains a pair of initial RPC authentication level settings and RPC authentication lower bound settings. One pair governs Cache Manager communications with File Servers in the same cell, while the second set governs communications with File Servers in foreign cells. Similarly, each File Server maintains a pair of RPC authentication lower and upper bound settings. Again, one pair governs communications with Cache Managers in the same cell, while the second pair controls communications with Cache Managers in foreign cells.

When a Cache Manager must contact a File Server to access a given fileset the Cache Manager and File Server negotiate for a mutually acceptable RPC authentication level. In operation, the process works as follows.

The Cache Manager sends an RPC to the File Server that is using the Cache Manager's initial RPC authentication level. The File Server checks the RPC and compares it to the authentication level range determined by the File Server's upper and lower authentication level bounds. If the RPC falls within the authentication level range, communications between the Cache Manager and File Server are established. However, if the RPC authentication level is above or below the File Server's range, the File Server responds with an instruction to increase or decrease the authentication level accordingly. This negotiation continues until the Cache Manager and File Server arrive at a mutually agreeable RPC authentication level or

until the File Server requests an authentication level below the minimum allowed for the Cache Manager (causing the Cache Manager to refuse communications with the File Server).

After arriving at a mutually agreeable RPC authentication level, the Cache Manager stores that information so that it does not need to renegotiate an authentication level during further communications with that particular file server.

Note: Cache Managers in versions of DFS earlier than OSF DCE 1.2.2 (for example, DCE for AIX 2.1) cannot negotiate RPC authentication levels. Setting the minimum authentication level bound at a File Exporter higher than packet prevents that File Server from communicating with Cache Managers based on earlier versions of DFS.

You can establish a Cache Manager's initial and lower bound RPC authentication levels by using the **dfsd** command. You must assume the **root** identity on the Cache Manager machine to issue this command. You can adjust these settings by using the **cm setprotectlevels** command. You can check the Cache Manager's current RPC authentication level settings with the **cm getprotectlevels** command.

You can establish the upper and lower File Exporter RPC authentication bounds by using the **fxd** command. You cannot display a File Exporter's RPC authentication bound settings. For more information about setting the File Exporter's authentication bounds with the **fxd** command, see the Reference part of this guide and reference.

DFS Distributed Database Technology

DFS includes two administrative databases: the Fileset Location Database (FLDB) and the Backup Database. You can increase system efficiency, file availability, and system reliability by replicating (copying) these two databases on multiple server machines. If one machine housing a copy of a database then becomes unavailable, the information can still be accessed from a copy of the database on another machine.

Unlike replicated filesets, replicated databases may change frequently. To ensure consistent system behavior, all copies of a database must be identical. DFS uses a library of utilities, *Ubik*, as a mechanism for synchronizing multiple copies of a replicated database. (Because Ubik is a subroutine library, it does not appear in listings of the processes running on a server machine.)

In DFS, one server machine houses a master copy of a replicated database such as the FLDB. When a user alters information in the database, Ubik coordinates the distribution of the change from the master copy to the copies of the database on other machines; the distribution is automatic and nearly instantaneous. Ubik dynamically selects a master copy of a database from among the servers that house it. The selection process and the propagation of changes to all copies of a database are managed entirely by Ubik and are transparent to administrators and users.

Ubik Database Synchronization

The Ubik library has a client portion and a server portion. Clients such as the **fts** and **bak** programs call subroutines in the Ubik library's client portion to contact the Fileset Location (FL) Server or Backup Server. These database server processes in

turn call subroutines in the server portion of the Ubik library to access or modify information in the FLDB or Backup Database.

The master copy of an FLDB or Backup Database is referred to as the *synchronization site*. The other copies of the database are referred to as *secondary sites*. A separate occurrence of Ubik, referred to as a *Ubik coordinator*, maintains the copy of the database at each site. A database server process makes a change to a database by issuing a call to the Ubik coordinator at the synchronization site, which makes the change to that copy of the database and distributes the change to the Ubik coordinators at the secondary sites. The coordinator at each secondary site then updates the copy of the database at its site.

Each copy of a database has a version number, which should always be the same for all copies of the database. Each change to a database increments the version number of the database by one. The coordinator at the synchronization site uses the version number to determine whether each secondary site has a copy of the most recent version of the database.

For example, if a service outage isolates a secondary site from the synchronization site, the secondary site no longer receives database updates from the synchronization site. When communications are restored, the coordinator at the synchronization site examines the version number of the database at the secondary site to determine whether the secondary site has the most recent version. If necessary, it sends the copy with the highest version number to the secondary site.

The Ubik coordinator at the synchronization site periodically sends an RPC to each secondary site. A response to the RPC from the coordinator at a secondary site serves as a *vote* to maintain the current synchronization site in its role for a fixed amount of time. Within that time, the synchronization site sends a subsequent RPC to the secondary site in an attempt to retain its role.

The coordinator at the synchronization site constantly tallies the votes it receives from the secondary sites. It continues in its role as synchronization site, confident that the other sites have not chosen a new synchronization site and begun making competing changes to the database, as long as it receives the votes of a strict majority (more than 50%) of all database sites, including itself. The necessary majority of database sites is referred to as a *quorum*.

Because Ubik relies on the actions of a quorum, having an odd number of database sites is helpful; in most cases, storing a replicated database at three sites is sufficient. Note, however, that the vote of the coordinator on the database server machine with the lowest network address of all database server machines of its type (those that house the FLDB or those that house the Backup Database) carries more weight than the votes of the coordinators at the other sites. This weighting allows Ubik to attain a quorum if an even number of sites exist.

The synchronization site stops sending RPCs to the secondary sites if hardware, software, or network problems result in any of the following:

- The synchronization site stops receiving votes from a quorum of the database sites.
- The synchronization site cannot propagate changes to a quorum of the database sites.
- The synchronization site or its machine fails.

If the coordinator at the synchronization site stops sending RPCs for any reason, Ubik elects a new synchronization site. In an election, each coordinator is biased to vote for the site with the lowest network address from among the sites it can contact. The vote of the site with the lowest network address of all database server machines of that type carries slightly more weight than the votes of the other sites. One site, usually the one with the lowest network address, typically gathers the necessary majority quickly and is elected the new synchronization site.

Immediately following the election, the newly elected synchronization site polls all sites to find the database with the highest version number. It adopts this version as the master copy and distributes it to the sites that do not yet have it. The election and database distribution are typically brief, usually taking no longer than a few minutes.

While Ubik cannot obtain quorum and during the subsequent election and database distribution, the affected database cannot be modified in any way. If the Backup Database is affected, information cannot be read from the database; the database is completely unavailable. If the FLDB is affected, Cache Managers can still read information from the database about the locations of filesets from which they need to access information; however, **fts** commands such as **fts lsfldb** cannot be used to get information from the database. Because the FLDB is most often accessed by Cache Managers seeking fileset location information, a Ubik election and ensuing database distribution do not interfere with the database's primary purpose.

Providing Information for Ubik

For the most part, Ubik operates without human intervention. However, it does depend on other DCE facilities and services for some things. The following list describes the interaction between Ubik and the remainder of DCE. It also provides an overview of the configuration information necessary for Ubik to operate properly. "Configuring Database Server Machines for Ubik" on page 59 discusses the database server configuration steps required for Ubik to function properly.

- *Ubik relies on DTS* to synchronize the clocks on server machines that house copies of a replicated database. Ubik coordinators must agree on the time; clock differences among Ubik sites can cause them to believe they are no longer in contact with each other, even if they are operating correctly. If a site falls out of touch, it may try to elect a new synchronization site or refuse to give out information. You can prevent such service outages by using DTS to synchronize the clocks on database server machines.
- *Ubik relies on the DCE Security Service* for secure communications between all Fileset Database machines (machines that house the FLDB) and Backup Database machines (machines that house the Backup Database). Each type of database server has its own security group, of which all machines that house a copy of that type of database must be members. A machine's membership in this group enables the Ubik coordinator on that machine to communicate with the Ubik coordinators on the other database servers of that type, thus allowing the coordinator to participate in Ubik elections.

  Abbreviated forms of the DFS server principals of all Fileset Database machines must be listed in the **subsys/dce/dfs-fs-servers** group in the Registry Database. Similarly, abbreviated forms of the DFS server principals of all Backup Database machines must be listed in the **subsys/dce/dfs-bak-servers** group in the Registry Database. To view the members of either of these security groups, use the **dcecp group list** command.

A machine's DFS server principal is of the form **/.../** *cellname***/hosts/** *hostname***/dfs-server**. The abbreviated form of a machine's DFS server principal is of the form **hosts/** *hostname***/dfs-server**. For example, in the cell named *abc.com*, the abbreviated server principals of all Fileset Database machines are listed in **subsys/dce/dfs-fs-servers** in the form **hosts/** *hostname***/dfs-server**.

- *Ubik relies on CDS* for a complete list of all Fileset Database and Backup Database machines. Each type of database server has its own RPC server group in CDS. Ubik examines the machines listed in the appropriate RPC group to determine how many sites constitute a majority and where to send votes in the event of an election.

  The names of the RPC bindings of all Fileset Database machines must be listed in the RPC group in CDS at **/.../** *cellname***/fs**, the junction to the DFS filespace. Likewise, the names of the RPC bindings of all Backup Database machines must be listed in the RPC group in CDS at **/.../** *cellname***/subsys/dce/dfs/bak**. To view the members of either of these RPC server groups, use the **dcecp rpcgroup list** command.

  The name of a machine's RPC binding is of the form **/.../** *cellname***/hosts/** *hostname***/self**. For example, in the cell named **abc.com**, the names of the RPC bindings of all Fileset Database machines are listed in **/.../abc.com/fs** in the form **/.../abc.com/hosts/** *hostname***/self**.

Note: In a server machine's DFS server principal or the name of its RPC binding, the element that follows the *cellname* component is not considered to be well known; for example, **hosts** could be **dfs-hosts**. However, the string used for the element must be applied consistently to all such names in a cell.

In addition, the names **/.../** *cellname***/fs** and **/.../** *cellname***/subsys/dce/dfs/bak** in CDS are not considered to be well known; either can be changed during installation and configuration of a cell. Conversely, the names of the **subsys/dce/dfs-fs-servers** and **subsys/dce/dfs-bak-servers** groups are well known and cannot be changed.

Configuring Database Server Machines for Ubik

A cell's initial database server machines are configured when DFS is installed and configured in the cell. If it becomes necessary to add or remove database server machines after initial cell configuration, you can use the **config.dfs** and **unconfig.dfs** commands or the AIX SMIT utility. For more information about the **config.dfs** and **unconfig.dfs** commands, see the Reference part of this guide and reference. For information about using the AIX SMIT utility, see the *IBM DCE for AIX, Version 2.2: Quick Beginnings*.

When the Cache Manager on a DFS client machine needs information from the FLDB in a cell, the **dfsbind** process on the machine provides it with information about the names and network addresses of the Fileset Database machines for the cell. The information is valid for a limited amount of time, 24 hours by default, after which the Cache Manager requests refreshed information from **dfsbind**; the Cache Manager also needs to refresh the information when it is restarted.

The **fxd** process on a File Server machine passes the same information about Fileset Database machines for the local cell to the File Exporter on its machine, but only when it is restarted (generally when the machine is rebooted).

It is seldom necessary to restart client or server machines if you reconfigure a cell's Fileset Database machines. As long as at least one Fileset Database machine remains the same after reconfiguration, all machines can continue to access the FLDB via that machine. Eventually, all machines recognize the current set of Fileset Database machines as a result of routine machine administration and maintenance. It is never necessary to restart client or server machines if you reconfigure a cell's Backup Database machines.

"Adding a Database Server Machine" and "Removing a Database Server Machine" on page 61 describe the steps required to add or remove a database server machine after initial cell configuration. Recall that each Fileset Database machine must run the FL Server (**flserver** process), and each Backup Database machine must run the Backup Server (**bakserver** process). These processes should be controlled by the Basic OverSeer (BOS) Server (**bosserver** process) on their machines, as recommended; if they are, you can use the appropriate **bos** commands to manipulate them.

Also recall that each FL Server must use the same **admin.fl** list and that each Backup Server must use the same **admin.bak** list. In addition, the abbreviated DFS server principal of each Fileset Database machine must be included in the **admin.fl** list, and the abbreviated DFS server principal of each Backup Database machine must be included in the **admin.bak** list. A DFS server principal can be added directly to a list, or it can be present as a member of a group included in the list (for example, the group **subsys/dce/dfs-fs-servers** can be included in the **admin.fl** list).

Inclusion in the appropriate administrative list allows the database server process at the synchronization site to distribute changes to the database server processes at the secondary sites. The Update Server should be used to propagate these administrative lists from the System Control machine to their respective database server machines.

You can use the **udebug** command to obtain status information on Ubik database servers. The command is useful for diagnosing problems associated with Ubik. (See the Reference part of this guide and reference for complete information about the **udebug** command and its options. Refer to the sections at the beginning of this chapter for more information about the processes that must run on either type of database server machine and the administrative lists used to specify who can control them; see "Chapter 5. Monitoring and Controlling Server Processes" on page 111 for more information about **bos** commands.)

Adding a Database Server Machine

Note: Instead of performing the following steps, you should use the **config.dfs** command or the AIX SMIT utility to add a database server.

To add a database server machine, do the following:

1. *If you intend to configure the machine as a Fileset Database machine* and the machine does not currently have a server entry in the FLDB, use the **fts crserverentry** command to create a server entry in the FLDB for the abbreviated DFS server principal of the machine. The machine already has a server entry in the FLDB if it is configured as a File Server machine. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for more information about using the **fts crserverentry** command to create server entries.)

2. Use the **dcecp group add** command to add the abbreviated DFS server principal of the new database server machine to the appropriate security group (**subsys/dce/dfs-fs-servers** or **subsys/dce/dfs-bak-servers**) in the Registry Database.
3. Use the **dcecp rpcgroup add** command to add the name of the RPC binding of the new database server machine to the appropriate RPC server group (**/.../** *cellname***/fs** or **/.../** *cellname***/subsys/dce/dfs/bak**) in CDS.
4. Use the **bos addadmin** command to add the abbreviated DFS server principal of the new database server machine to the appropriate administrative list (**admin.fl** or **admin.bak**). Doing so allows the synchronization site to propagate changes to the secondary sites. These administrative lists are usually updated on the cell's System Control machine, which then distributes the updated lists via the Update Server.

   Alternatively, you can use the **dcecp group add** command to add the abbreviated DFS server principal to a security group included in the list. Note that, if a group such as **subsys/dce/dfs-fs-servers** is included in the administrative list, the DFS server principal is already present in the list as a member of that group.
5. Copy the appropriate administrative list (**admin.fl** or **admin.bak**) to the */dcelocal***/var/dfs** directory on the new database server machine. These administrative lists are typically propagated from the cell's System Control machine via the Update Server. Modify the Update Server as necessary if the list is propagated from the cell's System Control machine.
6. Stop and restart the appropriate database server process (**flserver** or **bakserver**) on each database server machine of that type. Restarting the existing database server processes causes them to read the updated RPC server group, which ensures that each Ubik coordinator agrees on the number and identities of the other database server machines of its type. This agreement is vital to Ubik's use of a quorum of database server machines to maintain database consistency.
7. Start the appropriate database server process (**flserver** or **bakserver**) on the new database server machine.

Removing a Database Server Machine

Note: Instead of performing the following steps, you should use the **unconfig.dfs** command or the AIX SMIT utility to remove a database server.

To remove a database server machine, do the following:

1. Stop the appropriate database server process (**flserver** or **bakserver**) on the database server machine to be removed.
2. Use the **dcecp group remove** command to remove the abbreviated DFS server principal of the database server machine to be removed from the appropriate security group.
3. Use the **dcecp rpcgroup remove** command to remove the reference to the RPC binding of the database server machine to be removed from the appropriate RPC server group.
4. Use the **dcecp rpcentry show** command on each database server machine of the appropriate type to update the entry for the appropriate RPC server group from CDS. The command forces CDS to update information that it caches from the entry for the group in the namespace.
5. Stop and restart the appropriate database server process (**flserver** or **bakserver**) on each database server machine of that type. Restarting the

existing database server processes causes them to read the updated RPC server group, which ensures that each Ubik coordinator agrees on the number and identities of the other database server machines of its type. This agreement is vital to Ubik's use of a quorum of database server machines to maintain database consistency.

6. Use the **bos rmadmin** command to remove the abbreviated DFS server principal of the database server machine to be removed from the appropriate administrative list (**admin.fl** or **admin.bak**). These administrative lists are usually updated on the cell's System Control machine, which then distributes the updated lists via the Update Server.

   If you chose instead to add the abbreviated DFS server principal to a security group included in the list, you can use the **dcecp group remove** command to remove the server principal from the group. Note that if the DFS server principal was present in the administrative list as a member of a group such as **subsys/dce/dfs-fs-servers**, the server principal is already removed from the list.

7. Remove the appropriate administrative list (**admin.fl** or **admin.bak**) from the *dcelocal*/**var/dfs** directory on the database server machine to be removed. Modify the Update Server as necessary if the list is propagated from the cell's System Control machine.

Changing IP Addresses

To change the IP address of a machine that is configured as a CDS and a Security server, follow the procedure in the *IBM DCE for AIX, Version 2.2: Administration Guide* . If DFS or any DCE applications are running on the system, they need to be stopped before performing steps to change IP addresses for client machines. If the server is running as a DFS server, the FLDB server entry must be modified to reflect the new address. Details of these procedures are in the *IBM DCE for AIX, Version 2.2: Problem Determination Guide*.

Large File Support in DFS

Note: For AIX DFS, this support is only available if you are running DCE 2.2 for AIX on AIX 4.2.1 or higher.

DFS supports a maximum file size of greater than 2GB. To use files greater than 2GB, both the DFS client and server have to support large files. DFS clients and servers that do not support large files encounter errors attempting to access large files. The actual error returned to the application is dependent upon the level of DFS running on the platform and the particular vendor's implementation.

To ensure that the DFS fileset operations work properly when large files are present in the DFS filespace, the machine's default **ulimit** value for filesize should be set to **unlimited** prior to starting DFS on the machine.

DFS fileset operations such as move or replication fails if a fileset containing a large file is moved or replicated to a DFS server that does not support large files. If large files are used in a DCE/DFS environment, ensure that all DFS file server machines support large files to avoid unexpected errors during fileset operations.

Chapter 3. Using ACLs and Groups

This chapter summarizes the use of DCE Access Control Lists (ACLs) with DFS. DCE ACLs allow you to specify access to files and directories for individuals and groups of users. DCE ACLs can be used to protect files and directories stored in DCE LFS filesets. (See the Security Service portion of the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for details about manipulating DCE ACLs.)

This chapter also presents information about groups. In addition to using groups in ACLs, you can use groups in DFS administrative lists to specify the users who are allowed to issue commands that affect filesets and server processes. In this manner, you can precisely control the security of the administrative domains in your cell. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for complete details about using administrative lists; see the Security Service portion of the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for information about creating and maintaining groups.)

Note: The information in this chapter applies only to ACLs used with data stored in DCE LFS filesets. It does not apply to ACLs used with other DCE components. Differences exist between the use of DCE ACLs with DCE LFS objects and the use of DCE ACLs with other DCE components.

Using DCE ACLs with DFS

In the UNIX operating system, mode bits provide file system protection for file and directory objects (the general term ″object″ refers to a file or a directory). The access permissions for files and directories are set for three kinds of users: the user who owns the object, members of the group that owns the object, and all other users. The operations that these users can perform are determined by read, write, and execute mode bits.

All file and directory objects in DCE LFS filesets also have mode bits. However, the protection of such files and directories can be augmented with DCE ACLs, which allow access permissions to be defined for many different users and groups. With DCE ACLs, you can grant users six different permissions for your directories and four different permissions for your files. These permissions allow for the precise definition of access to directories and files.

DCE ACLs supplement the UNIX mode bits that are used to protect files and directories in DCE LFS filesets; they do not replace them. DCE LFS ensures that an object’s mode bits and its ACL permissions are always synchronized. Note that objects in DCE LFS filesets can rely exclusively on mode bits as their sole form of protection. (See “Initial Protection of a New File or Directory” on page 75 and “Initial ACLs of a New Fileset” on page 85 for more information about this possibility; see “ACL Interaction with UNIX Mode Bits” on page 74 for a description of the interaction and level of compatibility between DCE ACLs and UNIX mode bits.)

DCE ACLs are used only with objects in DCE LFS filesets. Mode bits are the only form of protection for objects in most non-LFS filesets.

ACL Entries

The DCE ACL for a file or directory object consists of multiple ACL entries. Each ACL entry defines the operations that a different user or group can perform on the object. Each entry has the following format:

{ *type* [*key*] *permissions*}

The elements of an entry provide the following information:

- The *type* specifies the kind of user or group to which the entry applies.
- The *key* names the specific user or group to which the entry applies. Some entries apply to predefined collections of users and so do not include a key.
- The *permissions* define the operations that can be performed on the object by the user or group to which the entry applies. ACLs on DCE LFS objects can include six access permissions: **r** (read), **w** (write), **x** (execute), **c** (control), **i** (insert), and **d** (delete).

An ACL entry is also used to define a mask that can be included on an ACL to limit the permissions granted by certain other entries. The following subsections provide more detailed information about the various ACL entry types and keys and the permissions they can grant.

Note: Although the text of this chapter refers primarily to ACL entries for users and groups, an ACL entry can apply to any principal (for example, to a server principal).

ACL Entry Types for Users and Groups

Most ACL entry types are used to specify the permissions granted to users and groups. To fully understand how ACL entries for users and groups are defined and interpreted, you need to understand the concept of an ACL's default cell. Recall that a user's local, or home, cell is the cell in whose Registry Database the user's principal and account are defined. Just as each user has a local cell, each ACL has a default cell.

An ACL's default cell names the cell with respect to which the ACL's entries are defined. A user or group named in an ACL entry is assumed to be from the default cell unless the entry explicitly names a different cell. The default cell is not necessarily the cell in which the ACL exists. For example, an object in cell **abc.com** can have an ACL whose default cell is **def.com**. With respect to ACLs, a local user is one whose local cell is the same as the default cell of an ACL; conversely, a foreign user is one whose default cell is different from the default cell of an ACL.

Table 4 lists the different types of ACL entries, their use of entry keys, and the users and groups to which they apply. As necessary, the table provides information about how an ACL's default cell affects the interpretation of the entry.

Table 4. ACL Entry Types for Users and Groups

Type	Key	Applies to
`user_obj`	None	The user who owns the object. The user is from the default cell.
`user`	*username*	The user *username* from the default cell.

Table 4. ACL Entry Types for Users and Groups (continued)

Type	Key	Applies to
foreign_user	*cell_name*/ *username*	The user *username* from the foreign cell *cell_name*.
group_obj	None	Members of the group that owns the object. The group is from the default cell.
group	*group_name*	Members of the group *group_name* from the default cell.
foreign_group	*cell_name*/ *group_name*	Members of the group *group_name* from the foreign cell *cell_name*.
other_obj	None	Users from the default cell who do not match any of the preceding entries.
foreign_other	*cell_name*	Users from the foreign cell *cell_name* who do not match any of the preceding entries.
any_other	None	Users from any foreign cell who do not match any of the preceding entries.

The default cell of an ACL, not the cell in which the ACL resides, determines the cell with respect to which the following entry types are defined:

- **user_obj**
- **user**
- **group_obj**
- **group**
- **other_obj**

For instance, a **user** entry specifies the permissions for a user whose local cell is the same as the default cell of an ACL. Whereas the entry types in the previous list refer to users and groups whose local cell is the same as an ACL's default cell, the **foreign_user**, **foreign_group**, **foreign_other**, and **any_other** entry types refer to users and groups whose local cells are different from an ACL's default cell. For instance, a **foreign_user** entry specifies the permissions for a user whose local cell is different from the default cell of an ACL. (Note that **foreign_** entries can exist for users or groups from the default cell. See "The Default Cell and ACL Inheritance" on page 76 for more information about an ACL's default cell, how it is listed, and how it is set.)

Some examples of ACL entries for users and groups follow:

{user_obj *permissions***}**
: Defines the permissions for the user who owns the object. The user is from the default cell.

{user frost *permissions***}**
: Defines the permissions for the user named **frost** from the default cell.

{group writers *permissions***}**
: Defines the permissions for the group named **writers** from the default cell.

{foreign_user /.../abc.com/wvh *permissions***}**
Defines the permissions for the user named **wvh** from the foreign cell named **abc.com**.

{foreign_group /.../abc.com/writers *permissions***}**
Defines the permissions for the group named **writers** from the foreign cell named **abc.com**.

The following rules govern the appearance of entries for users and groups on the ACLs of DCE LFS objects:

- The **user_obj**, **group_obj**, and **other_obj** entries must exist; all other entry types for users and groups are always optional.
- Only one entry of the same specificity (the same entry type and, if applicable, the same key) can exist on an ACL; for example, only one **user** entry can exist for a given *username* from the default cell.

Note: The first rule applies only to ACLs on DCE LFS objects, not to ACLs on objects associated with other DCE components. DCE LFS enforces these restrictions in an effort to track Draft 12 of the POSIX standard for ACLs on file and directory objects. (POSIX is a prominent collection of standards specifications for the computer industry.)

ACL Entry Types for Masks

DCE ACLs also provide a **mask_obj** entry type that can be used to filter, or mask, the permissions granted by certain user and group entries. The ACL **mask_obj** entry has the following format:

{**mask_obj** *permissions*}

The **mask_obj** entry specifies the maximum set of permissions that can be granted by any entries *except* the **user_obj** and **other_obj** entries. Permissions granted by any other entries are filtered through the **mask_obj**; only those permissions found in both the entry and the **mask_obj** are granted.

The **mask_obj** entry can only restrict the permissions granted by another entry; it cannot extend them. When DCE LFS determines the permissions granted to a user by an entry to which the **mask_obj** applies, it compares the permissions granted by the applicable entry with those permitted by the **mask_obj** entry. DCE LFS denies the user a permission granted by the applicable entry if the permission is not included in the permission set specified with the **mask_obj** entry. DCE LFS does not grant the user a permission specified with the **mask_obj** entry but not with the applicable entry.

If an entry other than **user_obj**, **group_obj**, or **other_obj** exists on an ACL, the **mask_obj** entry must exist as well. If a **mask_obj** entry does not already exist when an entry other than an **_obj** entry is created, the **dcecp acl** command, which is used to modify an ACL, automatically creates one. Note that the **mask_obj** entry filters the permissions granted to the **group_obj** entry, but an ACL can have a **group_obj** entry without having a **mask_obj** entry.

Note: The rule that requires the presence of the **mask_obj** entry with an entry other than an **_obj** entry applies only to ACLs on DCE LFS objects, not to ACLs on objects associated with other DCE components. DCE LFS enforces this restriction in an effort to track Draft 12 of the POSIX standard for ACLs on file and directory objects.

ACL Entry Types for Unauthenticated Users

An unauthenticated user is one whose DCE identity has not been verified by the DCE Security Service. For example, a user can access DCE without being authenticated by logging into the local machine without logging into DCE. In this case, the user is said to be unauthenticated because DCE cannot verify the user's identity. An authenticated user whose DCE credentials have expired is also considered an unauthenticated user.

When a user attempts to access an object, DFS first determines whether the user is authenticated. For access to an object in a DCE LFS fileset, an authenticated user acquires the permissions associated with the user's authenticated identity according to the normal ACL evaluation routine. (See "ACL Evaluation" on page 70 for a description.) An unauthenticated user's permissions are determined as follows:

1. DFS uses the identity **nobody** as the identity of the user; it treats the identity as an authenticated user from a nonexistent foreign cell.

   DFS assigns the identity **nobody** to all unauthenticated users, treating the identity as an authenticated user from an unknown foreign cell, regardless of the cell from which an unauthenticated user requests access to an object. DFS uses a fictitious cell as the local cell of the identity **nobody**; an entry for the fabricated cell cannot be created on an ACL. The primary group of the identity **nobody** is the group **nogroup**. The user ID of the identity **nobody** and the group ID of the group **nogroup** are both typically **-2**. However, these IDs can vary between File Server machines.
2. DCE LFS grants the user the permissions associated with the **any_other** entry.

   Because the user **nobody** is treated as a user from a *nonexistent* foreign cell, the user *cannot* match any **foreign_** entries (**foreign_user**, **foreign_group**, or **foreign_other**). The user is therefore granted the permissions associated with the **any_other** entry. If an **any_other** entry is not present on the ACL, the user has no permissions. To prevent unauthenticated users from acquiring permissions for an object, do not include an **any_other** entry on the object's ACL.

For access to objects in non-LFS filesets, unauthenticated users (regardless of their cells) and all foreign users (authenticated or unauthenticated) are treated as the user **nobody**. As a result, such users are granted the permissions associated with the **other** UNIX mode bits. Note that authenticated users from foreign cells are granted the permissions associated with their authenticated foreign identities when they access objects in DCE LFS filesets.

Note: DCE ACLs used with objects for other DCE components include an additional **unauthenticated** entry type that masks the permissions that can be granted to unauthenticated users. Prior to DCE 1.3 for AIX, DCE LFS allowed **unauthenticated** entries to be included on the ACLs of DCE LFS objects, but it ignored the entries when determining users' permissions.

As of DCE 1.3 for AIX, DCE LFS no longer allows **unauthenticated** entries to be included on the ACLs of DCE LFS objects. It is possible for the ACLs of existing objects to include **unauthenticated** entries added with earlier versions of DCE LFS. DCE LFS continues to ignore existing **unauthenticated** entries when determining permissions.

However, an ACL that includes an **unauthenticated** entry cannot be modified until the entry is removed from the ACL. An attempt to make any

other change to the ACL fails until the entry is removed. You can use the **dcecp acl modify** command with the **-remove** option to remove an **unauthenticated** entry from an ACL.

To prevent potential failures, you may want to remove **unauthenticated** entries from the ACLs of all DCE LFS objects. Moving or restoring a DCE LFS fileset to a File Server machine that is running DCE Version 1.1 or a later version of DCE automatically removes **unauthenticated** entries from the ACLs of all objects in the fileset. You can also write a script that removes the entries from the ACLs of all DCE LFS objects in your cell.

ACL Permissions

Each ACL entry for a user or group includes a set of permissions that defines the operations it grants to the user or users to whom it applies. For a **mask_obj** entry, the permissions define the maximum set of permissions that are allowed by the mask. Each entry can be assigned a different set of permissions.

The following permissions can be associated with an entry on an ACL for a file or directory in a DCE LFS fileset. All six permissions apply to a directory, but only the first four apply to a file; the insert and delete permissions are meaningless for files.

- **r** (read)
- **w** (write)
- **x** (execute)
- **c** (control)
- **i** (insert)
- **d** (delete)

The following table lists the various operations that can be performed on a file or directory and the ACL permissions that are required to perform them. All operations performed on a file or directory object require the **x** (execute) permission on each directory that leads to the object. Keep this requirement in mind when determining the permissions necessary to perform the operations described in the following chapters; not all operations list it explicitly.

Note: A user must have the **x** (execute) permission on each directory that leads to an object to access that object by its pathname. However, certain file system operations, such as the creation of hard links and mount points, can circumvent this restriction by supplanting the usual traversal of the pathname. To guarantee that an object is securely protected, set its permissions to the precise protections you want it to have. Do not rely on the absence of the **x** permission for a parent directory to prevent unwanted access of an object.

Table 5. File and Directory Operations and Required ACL Permissions

Operation	**Required Permissions**
Change to a directory	x on the directory itself x on all directories that lead to the directory
List the contents of a directory	r on the directory itself x on all directories that lead to the directory
List information about the objects in a directory	r and x on the directory itself x on all directories that lead to the directory

Table 5. File and Directory Operations and Required ACL Permissions (continued)

Operation	Required Permissions
Create an object	`w`, `x`, and `i` on the directory in which the object is to be placed `x` on all directories that lead to the directory in which the object is to be placed
Delete an object	`w`, `x`, and `d` on the directory from which the object is to be deleted `x` on all directories that lead to the directory from which the object is to be deleted
Rename an object	`w`, `x`, and `d` on the object's current directory `x` on all directories that lead to the object's current directory
	`w`, `x`, and `i` on the object's new directory `x` on all directories that lead to the object's new directory
	`w` on the object *if the object is a directory*
Read or read lock a file	`r` on the file itself `x` on all directories that lead to the file
Write or write lock a file	`w` on the file itself `x` on all directories that lead to the file
Execute a binary file	`x` on the file itself `x` on all directories that lead to the file
Execute a shell script	`r` and `x` on the script itself `x` on all directories that lead to the script
List the ACLs on an object	`x` on all directories that lead to the object
Change the ACLs on an object	`c` on the object itself `x` on all directories that lead to the object

Note: In the table above, the operation ″List the contents of a directory″ refers to displaying a simple list of the objects in a directory (for example, using the UNIX **ls** command with no flags or using the **ls -a** command). The operation ″List information about the objects in a directory″ refers to obtaining more detailed information about the objects in a directory, such as each object's mode bits or the time of its most recent update (for example, using the UNIX **ls -l** or **ls -t** command).

Also, if you rename an object and give it the name of an existing object, the object that exists with that name is deleted. In this case, you do not need the **d** permission on the parent directory of the existing object to delete that object.

For example, suppose the user **rajesh** needs to execute the DFS **fms** command. The command writes output to a log file named **FMSLog**, which it places in the directory from which it is issued. To create the file in a directory, **rajesh** must have the **w** (write), **x** (execute), and **i** (insert) permissions on the directory from which the command is issued, as well as the **x** (execute) permission on each directory that leads to the directory.

The following example ACL entry grants *rajesh* the **w**, **x**, and **i** permissions on the directory from which the command is issued. Each **-** (dash) indicates a permission that is not granted. Because a full permission set is **rwxcid**, this entry does not grant the **r** (read), **c** (control), and **d** (delete) permissions.

```
{user rajesh -wx-i-}
```

The following example ACL entry grants the user the execute permission on a directory that leads to the directory:

```
{user rajesh --x---}
```

ACL Evaluation

When a user tries to perform an operation on an object, DCE LFS examines the object's ACL to determine whether the user is granted the necessary permissions by an entry on the ACL. For example, to read a file, a user must be granted the read permission on the file (as well as the execute permission on each directory that leads to the file).

To determine a user's permissions for an object, DCE LFS evaluates the entries on the object's ACL according to the checking sequence described in the following list. DCE LFS stops evaluating the entries as soon as the user matches a condition described in the list. Evaluation proceeds to a condition in the checking sequence only if the user fails to match all of the previous conditions. (See Table 4 on page 64 for a description of the ACL entry types referred to in the following list.)

1. The user owns the object. DCE LFS grants the user the permissions specified with the **user_obj** entry. The permissions are *not* filtered through the **mask_obj** entry. Note that the **user_obj** entry always explicitly has the **c** permission; the **c** permission cannot be removed from the **user_obj** entry.
2. A **user** or **foreign_user** entry exists for the user. DCE LFS grants the user the permissions specified with the entry after filtering the permissions through the **mask_obj** entry.
3. The user belongs to the group that owns the object (the owning group's permissions are specified with the **group_obj** entry) or to any other groups that have **group** or **foreign_group** entries. If one or more group-related entries on the ACL apply, DCE LFS grants the user all of the permissions accrued from the applicable group entries after filtering the permissions through the **mask_obj** entry, if it exists. The user accrues permissions from all of the groups to which the user belongs.
4. The user is from the default cell. DCE LFS grants the user the permissions specified with the **other_obj** entry. The permissions are *not* filtered through the **mask_obj** entry.
5. The user belongs to a foreign cell that has a **foreign_other** entry. DCE LFS grants the user the permissions specified with the entry for that cell after filtering the permissions through the **mask_obj** entry.
6. The user is from a foreign cell that does not have a **foreign_other** entry. DCE LFS grants the user the permissions specified with the **any_other** entry, if it exists, after filtering the permissions through the **mask_obj** entry.
7. The user matches no entry. DCE LFS denies the user access to the object.

Before DCE LFS evaluates a user's permissions, DFS first determines whether the user is authenticated. If the user is authenticated, ACL evaluation proceeds as described in the previous list. If the user is not authenticated, DFS assigns the user the identity **nobody** and treats the identity as a foreign user from an unknown cell, regardless of the cell from which the unauthenticated user requests access to the object. ACL evaluation based on the identity **nobody** then proceeds accordingly.

When DCE LFS evaluates an ACL, it evaluates the more specific entries before it evaluates the less specific entries. Thus, the permissions granted to a group are applied to a user who is a member of the group only if the user is not granted permissions via the **user_obj** entry or a **user** or **foreign_user** entry. If an individual is granted one set of permissions as a user and another, wider set of permissions as a group member, the additional permissions granted to the group are not recognized; DCE LFS stops checking the ACL once it encounters the more specific user-related entry.

For example, suppose user **dale** belongs to a group that has the read and write permissions on a file through the **group_obj** entry on the file's ACL. Suppose further that **dale** is also specified in a **user** entry that grants only the read permission. The relevant entries from the ACL follow (assume the **mask_obj** entry permits the **r** and **w** permissions):

```
{user dale r-----}
{group_obj rw----}
```

Because the more specific **user** entry is evaluated before the **group_obj** entry, DCE LFS denies **dale** write access for the file.

Note: A user can match both the **user_obj** entry and a **user** or **foreign_user** entry; in this case, the user is granted permissions from only the **user_obj** entry. Similarly, a group can match both the **group_obj** entry and a **group** or **foreign_group** entry; in this case, however, members of the group accrue permissions from both entries.

ACL Evaluation for Local Access

If the **mount** command or its equivalent has been configured properly on your operating system, you can mount a DCE LFS fileset locally, as a file system on its File Server machine. You can access an object in a locally mounted DCE LFS fileset via a local pathname, as well as via a DCE pathname. In either case, the same ACL evaluation algorithm determines your permissions. However, if your local identity is different from your DCE identity, the permissions you receive when you access the object via its local pathname are those associated with your local identity, but the permissions you receive for access via the object's DCE pathname are those associated with your DCE identity. This is also true of objects in non-LFS filesets.

For example, suppose you log into the local machine as the **root** user and then authenticate to DCE as your DCE identity. If you access an object via its local pathname, you receive **root** permissions for the object; if you access the same object via its DCE pathname, you receive the permissions associated with your DCE identity.

If you log into the local machine as **root** without authenticating to DCE, you assume the identity of the local machine's ***/.../*** *cellname***/hosts/** *hostname***/self** principal for DCE access. If you access an object via its local pathname, you receive **root** permissions; however, if you access the object via its DCE pathname, you receive the permissions associated with the **self** identity of the local machine. To allow processes running as **root** on a machine (for example, **cron** jobs) to access an object via its DCE pathname, you can include an entry for the machine's **self** identity on the ACL of the object. The **self** identity can also receive permissions from a group to which it belongs or from the **other_obj** entry (because it is treated as an authenticated user from the local cell).

Setting and Examining ACLs

The **dcecp acl** command is used to list and modify the ACLs of DCE LFS objects. In most respects, the operation of the **dcecp acl** command with DCE LFS objects parallels its use with other types of DCE objects, as follows:

- To list the entries on an ACL for a file or directory, use the **dcecp acl show** command. To list an object's ACL, you must have the **x** (execute) permission on the directory in which the object resides, as well as on all directories that lead to that directory.

- To modify an entry on an ACL for a file or directory, use the **dcecp acl modify** command with one of the following options: **-add**, **-change**, **-remove**, or **-purge**. You can also use the **dcecp acl delete** and **dcecp acl replace** commands to modify an ACL. To modify an object's ACL, you must have the **c** (control) permission for the object, as well as the **x** (execute) permission on each directory that leads to the object.

Because the **user_obj** entry always has the control permission, you can always modify the ACL of an object that you own (an object for which the **user_obj** entry applies to you). To determine if you have the control permission for an object that you do not own, use the **dcecp acl show** command to display the object's ACL. You can also use the **dcecp acl check** command to display your permissions for an object.

The following subsections provide information about modifying ACLs on DCE LFS objects, including brief examples of using the **dcecp acl** command to list and modify a directory's ACL. (See the Security Service portion of the*IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for complete details about using the command to set and examine an object's ACLs.)

Rules for Modifying ACLs

A number of rules restrict the changes you can make to the ACL of a file or directory in a DCE LFS fileset. The following rules apply only to DCE LFS file and directory objects:

- The **user_obj**, **group_obj**, and **other_obj** entries must always exist. All other entry types are optional.
- The **mask_obj** entry must exist if an entry other than **user_obj**, **group_obj**, or **other_obj** exists. If the **mask_obj** entry does not exist when an entry other than an **_obj** entry is created, the **dcecp acl** command automatically creates it.
- The **user_obj** entry must always explicitly retain the **c** permission. This requirement prevents the owner of an object from being denied access to it; the owner can always grant himself or herself additional permissions.

These rules restrict your use of the **dcecp acl** command. Namely, if a single **dcecp acl modify** command modifies an ACL in a way that violates any of these restrictions, the command must include additional changes that reinstate the necessary entries or permissions. A single instance of the **dcecp acl modify** command cannot be used to effect a set of changes that violates these restrictions. The **dcecp acl delete** and **dcecp acl replace** commands can also never be used to violate these restrictions. The **dcecp acl** command makes changes to an ACL in the order in which the changes are specified on the command line; for a given **dcecp acl** command and a given ACL, either all of the changes indicated by the command are applied or none of the changes are applied.

Finally, recall that only one entry of the same specificity can exist on an ACL; for example, only one **user** entry can exist for a given *username*. Permissions you assign to an entry when you issue the **dcecp acl modify** command with the **-change** option *replace* the existing permissions associated with the entry; the specified permissions are not added to the existing permissions. If you want a user or group to retain the permissions already granted, you must include those permissions with the entry that you specify with the command. (Note that a **dcecp acl modify** command that includes the **-add** option fails if the entry to be added exists on the ACL.)

Examples of Listing and Modifying an ACL

The examples in this section demonstrate the use of the **dcecp acl** command to list and modify a directory's ACL. The following example uses the **dcecp acl show** command to display an object's ACL. The example shows the output of the command when it is used to display the ACL for the directory **drafts**:

```
dcecp> acl show /.../abc.com/fs/doc/drafts
```

```
{mask_obj r-x---}
{user_obj rwxcid}
{user dale rwx-id effective r-x---}
{group_obj rwx--- effective r-x---}
{group writers rwx--- effective r-x---}
{other_obj rwx---}
```

The output displays the ACL entries for the object. If an entry's permissions are restricted by the **mask_obj** entry, the permissions that remain after filtering through the mask are labeled **effective**. In this example, the permissions (**rwx-id**) granted to **dale** are restricted by the **mask_obj** entry to **r** and **x**. Users belonging to the group **writers** are, like **dale**, restricted to **r** and **x** access. The owner of the directory (**user_obj**) retains all of the specified permissions (**rwxcid**) because **user_obj** is not filtered by **mask_obj**.

Suppose another user, **pierette**, needs to have all of the permissions except **c** on the directory. Suppose further that **pierette** is a member of the group **writers**, which effectively has only the **r** and **x** permissions on the directory. To give **pierette** the required permissions, the following need to be done:

- A **user** entry for **pierette** needs to be added to grant the desired permissions, not all of which are granted to the group **writers**.
- The **mask_obj** entry needs to be expanded to allow for the additional permissions; it currently filters all user and group entries to only the **r** and **x** permissions.

The following example performs both operations with one invocation of the **dcecp acl modify** command. It uses the **-add** option to add an entry for **pierette** to the ACL. It also uses the **-mask** option with the value **calc** to recalculate the permissions granted by the **mask_obj** entry to include those to be granted to **pierette**. Alternatively, the **-mask** option could be used with the value **nocalc** to prevent recalculation of the permissions granted by the **mask_obj** entry, but doing so would cause the **mask_obj** entry to restrict **pierette**'s permissions. The command fails unless one of the two values is specified with the **-mask** option.

Note: The **dcecp acl modify** command dynamically recalculates the **mask_obj** entry as necessary when new entries are added to an ACL. By default, it refuses to readjust the **mask_obj** entry if doing so would grant currently masked permissions to another entry. In such cases, you must specify the **calc** or **nocalc** value with the **-mask** option to direct the command's actions with respect to the **mask_obj** entry.

```
dcecp> acl modify /.../abc.com/fs/doc/drafts -add {user pierette rwxid} \
> -mask  calc
```

The following example displays the new and modified ACL entries that grant **pierette** all permissions except **c**. Note that expanding the permissions allowed by the **mask_obj** entry increased the permissions granted to the other entries filtered by the mask.

```
dcecp> acl show /.../abc.com/fs/doc/drafts
```

```
{mask_obj rwx-id}
{user_obj rwxcid}
{user dale rwx-id}
{user pierette rwx-id}
{group_obj rwx---}
{group writers rwx---}
{other_obj rwx---}
```

Recall that DCE LFS evaluates the more specific **user** entries before it checks the less specific entries. Therefore, **pierette**, although a member of the group **writers**, receives the permissions granted by the **user pierette** entry. This is true regardless of whether **pierette** is granted more or fewer permissions via the **user** entry.

ACL Interaction with UNIX Mode Bits

In the UNIX file system, every file and directory object has associated with it a set of mode bits that provide information about the object. In addition to identifying the type of the object (file or directory), the bits define the permissions granted to three types of users: the user who owns the object, members of the group that owns the object, and all other system users. These mode bits are referred to as the **user**, **group**, and **other** mode bits, respectively.

Each type of user (**user**, **group**, and **other**) can be assigned any combination of the **r**, **w**, and **x** permissions via the appropriate mode bits. The operations associated with the bits are similar to those associated with the same permissions for DCE ACLs. The mode bits for an object can be listed with the UNIX **ls -l** command or its equivalent; they can be set with the UNIX **chmod** command or its equivalent.

Because DCE ACLs can be used only with objects in DCE LFS filesets, mode bits are the only form of protection associated with objects in most non-LFS filesets. In DCE LFS filesets, all file and directory objects can have both UNIX mode bits and DCE ACLs. Note that all objects always have UNIX mode bits, but they do not necessarily have ACLs. (See “Initial Protection of a New File or Directory” on page 75 for more details.)

For DCE LFS objects, DCE LFS synchronizes the protections set by an object’s UNIX mode bits with the protections set by its DCE ACL. It maintains symmetry between an object’s mode bits and its ACL permissions as follows:

- The **user** mode bits are identified with the **r**, **w**, and **x** permissions of the **user_obj** entry.
- The **other** mode bits are identified with the **r**, **w**, and **x** permissions of the **other_obj** entry.
- The **group** mode bits are identified with the **r**, **w**, and **x** permissions of the **mask_obj** entry. If the **mask_obj** entry does not exist (which is the case with the root directory of a newly created DCE LFS fileset, for example), the **group** mode bits are identified with the **r**, **w**, and **x** permissions of the **group_obj** entry. If the mode bits correspond to the **mask_obj** entry, they do not correspond to the **group_obj** entry, and vice versa.

To maintain this correspondence, when you modify an ACL **_obj** entry (**user_obj**, **mask_obj** or **group_obj**, or **other_obj**), DCE LFS updates the corresponding UNIX mode bits (**user**, **group**, or **other**) to reflect the permissions associated with the **_obj** entry. For example, suppose a file’s ACL has the following entries:

```
{mask_obj r-----}
{user_obj rwxc--}
{group_obj r-x--- effective r-----}
{other_obj ------}
```

DCE LFS sets the corresponding UNIX mode bits for the file to make the **user** mode bits **r**, **w**, and **x** and the **group** mode bits **r**. These mode bits are displayed with the **ls -l** command as follows:

```
-rwxr----- 1 dale       3625 Nov 22 11:36 filename
```

Suppose you then use the **dcecp acl modify** command to modify the ACL to give the **other_obj** entry the **r**, **w**, and **x** permissions (leaving the other entries unchanged), as follows:

```
{mask_obj r-----}
{user_obj rwxc--}
{group_obj r-x--- effective r-----}
{other_obj rwx---}
```

DCE LFS adjusts the UNIX mode bits to make the **other** mode bits **r**, **w**, and **x** in accordance with the **other_obj** entry. Displayed with the **ls -l** command, the mode bits are now as follows:

```
-rwxr--rwx 1 dale       3625 Nov 22 11:36 filename
```

Similarly, if you use the UNIX **chmod** command to modify the mode bits associated with an object, DCE LFS reconciles the corresponding ACL entries accordingly. Thus, DCE LFS ensures that the mode bits and ACL permissions of an object always agree.

It is worth noting that for an executable file (for example, a binary file) to be executed, the **x** mode bit must be assigned to one or more of **user**, **group**, or **other**. If one of these sets of mode bits does not include the **x** mode bit, no one can execute the file, not even the **root** user. For an executable file that has an ACL, at least one of the following ACL entries must have the **x** permission for the file to be executed: **user_obj**, **mask_obj** (or **group_obj**, if the **mask_obj** entry does not exist on the ACL), or **other_obj**.

Initial Protection of a New File or Directory

Each DCE LFS file or directory can have an Object ACL that controls access to the file or directory; all previous examples in this chapter refer to the Object ACL. Because they can contain other objects, directories (also referred to as *container objects*) can have two additional ACLs that determine the default ACLs to be inherited by objects created in them. Thus, a directory can have the following three ACLs:

Object ACL

Controls access to the directory itself. By default, this is the ACL that the **dcecp acl** command displays or modifies when it is issued.

Initial Object Creation ACL

Determines the default ACL inherited by files created in the directory. To view or modify a directory's Initial Object Creation ACL, include the **-io** option with the **dcecp acl** command.

Initial Container Creation ACL

Determines the default ACL inherited by subdirectories created in the directory. To view or modify a directory's Initial Container Creation ACL, include the **-ic** option with the **dcecp acl** command.

A directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL can exist independently of one another; they do not need to exist at all. A given directory can have all, some, or none of these ACLs. The type of file system protection (ACLs or UNIX mode bits) used initially for a new file or directory object depends on whether the parent directory of the new object has the appropriate Initial Creation ACL, as follows:

- If a new object's parent directory has the appropriate Initial Creation ACL, the new object inherits an Object ACL as its form of protection. The new object also has mode bits, but the Object ACL supplements these bits. Recall that DCE LFS ensures that the object's mode bits and its ACL permissions are always synchronized.
- If a new object's parent directory does *not* have the appropriate Initial Creation ACL, the new object initially has no Object ACL; the object relies on mode bits as its only form of protection. If they are not inherited, ACLs can be explicitly created with the **dcecp acl** command. (See "Mode Bits for New Objects That Do Not Inherit ACLs" on page 83 for information about using the **dcecp acl** command to create a directory's initial ACLs.)

  Note: An Object ACL is always created for a file or directory that is created by a foreign user, even if the parent directory does not have the appropriate Initial Creation ACL. (See "ACL Inheritance for Objects Created by Foreign Users" on page 80 and "Mode Bits for New Objects That Do Not Inherit ACLs" on page 83 for more information.)

The following subsections describe how the initial protections of a new object are derived. The first subsection provides more information about an ACL's default cell, which plays an important role in determining ACL inheritance. The following subsections describe ACL inheritance for objects created by local users and objects created by foreign users; both of these subsections assume that the parent directory has the appropriate Initial Creation ACL. The final subsection discusses how the UNIX mode bits are determined for an object whose parent directory does not have the appropriate Initial Creation ACL.

The Default Cell and ACL Inheritance

Recall that an ACL's default cell names the cell with respect to which the ACL is defined, and the default cell is not necessarily the cell in which the ACL exists. For example, an object in cell **abc.com** can have an ACL whose default cell is **def.com**. In this case, even though the object resides in cell **abc.com**, users from cell **abc.com** are foreign users with respect to the object's ACL.

With respect to ACLs, local users and foreign users are defined in terms of an ACL's default cell as follows:

- A local user is one whose local cell is the same as the default cell of an ACL. The following entry types are defined for local users and groups:
  - **user_obj**
  - **group_obj**
  - **other_obj**
  - **user**
  - **group**

For example, an entry of the type **user** *username* specifies the permissions for the user *username* whose local cell is the same as the default cell of the ACL.

- A foreign user is one whose local cell is different from the default cell of an ACL. The following entry types are defined for foreign users and groups:
  - **foreign_user**
  - **foreign_group**
  - **foreign_other**
  - **any_other**

  For example, an entry of the type **foreign_user** *cell_name username* specifies the permissions for the user *username* from the cell *cell_name*. (The *cell_name* of a **foreign_** entry is usually different from the default cell of an ACL, but it does not have to be.)

A directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL each have their own default cell. When a file or directory object is initially created, the default cell of its Object ACL is set to the local cell of the user who creates the object (the object's owner, who is named with the **user_obj** entry). The default cells of a new directory's Initial Creation ACLs are also set to the local cell of the user who creates the directory.

Listing an ACL's Default Cell: To determine the default cell of an ACL, include the **-cell** option with the **dcecp acl show** command, as follows:

```
dcecp> acl show pathname -cell
/.../cell_name
```

For example, the output for an ACL whose default cell is **abc.com** is the following:

```
/.../abc.com
```

Changing an ACL's Default Cell: To change the default cell of an ACL, use the **-cell** option with the **dcecp acl modify** command. If you indicate multiple changes with the command, the change to the default cell is applied before any other changes are applied.

The default cell of the Object ACL for an object can be changed only by a cell administrator for the File Server machine on which the object resides. The default cell of an Initial Creation ACL for a directory can be changed by any user who has the **c** permission on the directory's Object ACL, which always includes the owner of the directory, or by a cell administrator for the File Server machine on which the object resides. (Cell administrators are members of the group specified with the **-admingroup** option of the **fxd** command issued on the File Server machine.)

Although you can make the default cells of a directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL different from one another, this practice is not recommended. Changing each of a directory's ACLs to have different default cells can make it difficult to predict the effects of ACL inheritance. Also, because the default cell of an object's Object ACL is determined by the local cell of the user who creates the object, not by the default cell of the Initial Creation ACL that the object inherits, changing the default cell of an Initial Creation ACL is of limited utility.

Note that changing an ACL's default cell by including the **-cell** option with the **dcecp acl modify** command changes the scope of the ACL's entries. For example, the **other_obj** entry no longer applies to users from the former default cell; it now

applies to users from the new default cell. Also, entry types such as **user_obj** and **user**, which are defined with respect to an ACL's default cell, can now give permissions to different users in the new default cell.

If you change the default cell of an ACL, make sure you also change any **user** and **group** entries on the ACL to **foreign_user** and **foreign_group** entries as necessary. You may also want to change any **foreign_user** and **foreign_group** entries that apply to the new default cell to **user** and **group** entries.

ACL Inheritance for Objects Created by Local Users

For ACLs, a local user is a user whose local cell is the same as the default cell of an ACL. When a local user creates an object in a directory that has the appropriate Initial Creation ACL, DCE LFS uses the intersection of the following information to determine the Object ACL that it creates for the new object:

- The UNIX mode bits specified at the system call level (with the UNIX **open( )**, **creat( )**, or **mkdir( )** system call) when the object is created. The application that invokes one of these system calls specifies the mode bits for the new object. For example, when the UNIX **touch** command is used to create an object, the command usually specifies the **user**, **group**, and **other** mode bits as **r** and **w** in the resulting **creat( )** system call.
- The appropriate Initial Creation ACL of the object's parent directory. The parent's Initial Object Creation ACL is used for a file; the parent's Initial Container Creation ACL is used for a directory.

For example, when a file is created, DCE LFS derives the initial ACL entries and permissions for its Object ACL, the only ACL associated with a file, as follows:

- The **r**, **w**, and **x** permissions for the file's **user_obj** entry consist of the intersection of the **user** mode bits specified when the file is created and the corresponding permissions of the **user_obj** entry of its parent directory's Initial Object Creation ACL. The **c**, **i**, and **d** permissions for the file's **user_obj** entry are copied directly from the **user_obj** entry of the parent's Initial Object Creation ACL.
- The **r**, **w**, and **x** permissions for the file's **mask_obj** entry consist of the intersection of the **group** mode bits specified when the file is created and the corresponding permissions of the **mask_obj** entry of its parent directory's Initial Object Creation ACL. The **c**, **i**, and **d** permissions for the file's **mask_obj** entry are copied directly from the **mask_obj** entry of the parent's Initial Object Creation ACL. In addition, the **group_obj** entry is copied directly from the parent's Initial Object Creation ACL to the file's Object ACL.

  If the **mask_obj** entry does not exist on the parent's Initial Object Creation ACL, the **r**, **w**, and **x** permissions for the file's **group_obj** entry are defined as the intersection of the **group** mode bits specified when the file is created and the corresponding permissions of the **group_obj** entry of its parent directory's Initial Object Creation ACL. The **c**, **i**, and **d** permissions for the file's **group_obj** entry are copied directly from the **group_obj** entry of the parent's Initial Object Creation ACL.
- The **r**, **w**, and **x** permissions for the file's **other_obj** entry consist of the intersection of the **other** mode bits specified when the file is created and the corresponding permissions of the **other_obj** entry of its parent directory's Initial Object Creation ACL. The **c**, **i**, and **d** permissions for the file's **other_obj** entry are copied directly from the **other_obj** entry of the parent's Initial Object Creation ACL.
- All other entries included on the parent directory's Initial Object Creation ACL are copied directly to the file's Object ACL.

DCE LFS uses the same algorithm to determine the initial entries and permissions for a subdirectory's Object ACL, but it uses the parent directory's Initial Container Creation ACL instead of its Initial Object Creation ACL. The subdirectory also inherits its parent's Initial Container Creation ACL as its Initial Container Creation ACL, and it inherits its parent's Initial Object Creation ACL as its Initial Object Creation ACL. The subdirectory inherits these Initial Creation ACLs unchanged from its parent directory.

Figure 5 illustrates ACL inheritance for files and directories.

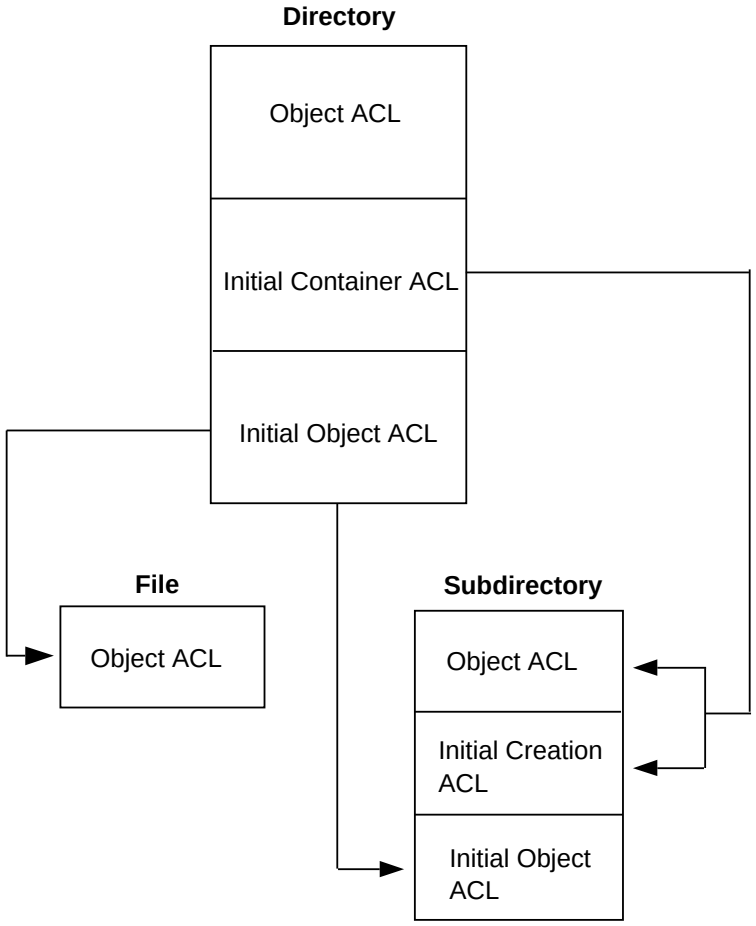


Figure 5. ACL Inheritance

The following simple example demonstrates ACL inheritance. In the example, the directory **/.../abc.com/fs/usr/rajesh** is the home directory for the user **rajesh**, whose local cell, **abc.com**, is the same as the default cell of the directory's ACLs. The following **dcecp acl show** command displays the Object ACL of the directory:

```
dcecp> acl show /.:/fs/usr/rajesh
{mask_obj rwx-id}
{user_obj rwxcid}
{user vijay rwx-id}
{group_obj r-x---}
{other_obj r-x---}
```

The following **dcecp acl show** commands show the Initial Object Creation ACL and Initial Container Creation ACL of the directory:

```
dcecp> acl show /.:/fs/usr/rajesh -io
{mask_obj rw----}
{user_obj rw-c--}
{user pierette rw----}
{group_obj r-----}
{other_obj r-----}
dcecp> acl show /.:/fs/usr/rajesh -ic
{mask_obj rwx-id}
{user_obj rwxcid}
{user pierette rwx-id}
{group_obj r-x---}
{other_obj r-x---}
```

Suppose **rajesh**, the owner of the directory, creates a subdirectory named **myfiles** in the directory. As the owner of the parent directory, **rajesh** is granted the permissions associated with the **user_obj** entry of the parent's Object ACL. The **user_obj** entry includes the **w**, **x**, and **i** permissions, so **rajesh** can create objects in the directory.

The **user_obj**, **mask_obj**, and **other_obj** permissions of the Object ACL for the new **myfiles** subdirectory are derived from the intersection of the permissions granted to these entries in the parent directory's Initial Container Creation ACL and the **user**, **group**, and **other** mode bits specified when the subdirectory is created. If the **user**, **group**, and **other** mode bits are all **r**, **w**, and **x** in the system call that creates the **myfiles** subdirectory, the subdirectory inherits the following Object ACL:

```
dcecp> acl show /.:/fs/usr/rajesh/myfiles
{mask_obj rwx-id}
{user_obj rwxcid}
{user pierette rwx-id}
{group_obj r-x---}
{other_obj r-x---}
```

Because the Initial Container Creation ACL includes a **mask_obj** entry, the **myfiles** subdirectory inherits the **group_obj** entry directly from the Initial Container Creation ACL. Similarly, the subdirectory inherits the **user** entry for **pierette** directly from the Initial Container Creation ACL. The subdirectory also inherits the Initial Container Creation ACL and Initial Object Creation ACL unchanged from its parent directory.

Note: An object's existing ACLs may not be maintained across a file system operation such as a move or copy (performed with the **mv** and **cp** commands in the UNIX operating system). Refer to your vendor's documentation for information about how ACLs are treated with respect to such operations.

ACL Inheritance for Objects Created by Foreign Users

For ACLs, a foreign user is a user whose local cell is different from the default cell of an ACL. Any user who has the **w**, **x**, and **i** permissions on a directory's Object ACL can create objects in the directory, regardless of whether the user is a foreign user with respect to the directory's ACL. For example, a user from the cell **def.com** who has the **w**, **x**, and **i** permissions on a directory whose default cell is **abc.com** can create an object in the directory. The default cell of the new object is **def.com**, not **abc.com**.

When a foreign user creates an object, ACL inheritance occurs as described in "ACL Inheritance for Objects Created by Local Users" on page 78. However, if the

user is a foreign user with respect to the appropriate Initial Creation ACL, entries inherited from the Initial Creation ACL are modified as follows:

- The **mask_obj** entry remains unchanged. It applies to the same entries on both the Initial Creation ACL and the new Object ACL.
- The **user_obj**, **group_obj**, and **other_obj** entries remain unchanged, but they are defined with respect to the default cell of the new Object ACL, not the default cell of the Initial Creation ACL. The **user_obj** entry specifies the permissions granted to the user who creates the object (the user whose local cell dictates the default cell of the ACL).
- Any **user** and **group** entries are changed to **foreign_user** and **foreign_group** entries because they are not defined with respect to the default cell of the new Object ACL.
- Any **foreign_user** and **foreign_group** entries that are defined with respect to the default cell of the new object ACL are changed to **user** and **group** entries.
- Any **foreign_user** and **foreign_group** entries that are defined with respect to neither the default cell of the Initial Creation ACL nor the default cell of the new Object ACL remain unchanged.
- Any **foreign_other** entries and the **any_other** entry remain unchanged.

If a foreign user creates an object in a directory, an Object ACL is created for the new object even if the parent directory does not have the appropriate Initial Creation ACL. In this case, the Object ACL must be created to record the fact that the new object's default cell is different from the cell in which the object resides. Because an unauthenticated user is treated as a user from an unknown foreign cell, an Object ACL is always created for an object created by an unauthenticated user also. (See "ACL Inheritance for Objects Created by Foreign Users" on page 80 for information about how the permissions granted by such an Object ACL are determined.)

The following example demonstrates what happens when the local cell of a user who creates an object is different from the default cell of the appropriate Initial Creation ACL of the directory in which the object is created. In the example, the directory **/.../abc.com/fs/usr/srivas** is the home directory of the user **srivas**, whose local cell, **abc.com**, is the same as the default cell of the directory's ACLs. The following **dcecp acl show** command displays the Object ACL of the directory:

```
dcecp> acl show /.:/fs/usr/srivas

{mask_obj rwx-id}
{user_obj rwxcid}
{user vijay rwx-id}
{foreign_user /.../def.com/andi rwx-id}
{foreign_user /.../ghi.com/pervaze r-x---}
{group_obj r-x---}
{other_obj r-x---}
{foreign_other /.../def.com r-x---}
```

The following **dcecp acl show** commands display the Initial Object Creation ACL and Initial Container Creation ACL of the directory:

```
dcecp> acl show /.:/fs/usr/srivas -io

{mask_obj rw----}
{user_obj rw-c--}
{user pierette rw----}
{foreign_user /.../def.com/andi rw----}
{foreign_user /.../ghi.com/pervaze r-----}
{group_obj r-----}
{other_obj r-----}
{foreign_other /.../def.com r-----}
```

```
dcecp> acl show /.:/fs/usr/srivas -ic
{mask_obj rwx-id}
{user_obj rwxcid}
{user pierette rwx-id}
{foreign_user /.../def.com/andi rwx-id}
{foreign_user /.../ghi.com/pervaze r-x---}
{group_obj r-x---}
{other_obj r-x---}
{foreign_other /.../def.com r-x---}
```

All three of these ACLs are defined with respect to the cell **abc.com**. For example, the **user** entries for **pierette** and **vijay** apply to specific users from the cell **abc.com**, and the **other_obj** entries apply to other users from the cell **abc.com**.

The user **andi**, who is from the cell **def.com**, has entries on all three of the directory's ACLs. The **foreign_user** entry on the Object ACL allows **andi** to create objects in the directory. If **andi** creates a subdirectory named **andi_files** in the directory, the default cell of the subdirectory is **def.com**. Assuming the **user**, **group**, and **other** mode bits are **r**, **w**, and **x** in the system call that creates the subdirectory, the Object ACL of the subdirectory inherits the following entries from the Initial Container Creation ACL of the parent directory:

```
dcecp> acl show /.:/fs/usr/srivas/andi_files
{mask_obj rwx-id}
{user_obj rwxcid}
{user andi rwx-id}
{foreign_user /.../abc.com/pierette rwx-id}
{foreign_user /.../ghi.com/pervaze r-x---}
{group_obj r-x---}
{other_obj r-x---}
{foreign_other /.../def.com r-x---}
```

The permissions granted by the various entries are inherited according to the ACL inheritance algorithm. However, because **andi**'s local cell (**def.com**) is different from the default cell (**abc.com**) of the parent directory's Initial Container Creation ACL, the entries from the parent's Initial Container Creation ACL are interpreted and modified as follows for use on the Object ACL of the **andi_files** subdirectory:

- The **mask_obj** entry is unchanged because it applies to the same users on both ACLs.
- The **user_obj**, **group_obj**, and **other_obj** entries are unchanged, but they now apply to users and groups from the cell **def.com**. The **user_obj** entry grants permissions to the user **andi**.
- The **user pierette** entry is changed to the **foreign_user /.../abc.com/pierette** entry because it is no longer defined with respect to the default cell of the ACL.
- The **foreign_user /.../def.com/andi** entry is changed to the **user andi** entry because it is now defined with respect to the default cell of the ACL. Note that, as the owner of the directory, **andi** derives permissions from the **user_obj** entry, so the **user** entry for **andi** is not used. It remains on the ACL nonetheless.
- The **foreign_user /.../ghi.com/pervaze** entry is unchanged because it is defined with respect to neither the default cell of the Initial Container Creation ACL of the parent directory nor the Object ACL of the new directory.
- The **foreign_other /.../def.com** entry is unchanged; it continues to apply to users from the cell **def.com**. However, because the default cell of the ACL is now **def.com**, users from that cell who are not granted permissions from specific user or group entries are now granted permissions from the **other_obj** entry. The **foreign_other** entry for users from the cell **/.../def.com** remains on the ACL, but

as long as the default cell of the ACL is **def.com**, this **foreign_other** entry does not determine the permissions granted to users from the **def.com** cell.

Note: You can explicitly include **foreign_other** entries for the default cell on a directory's Initial Creation ACLs to grant users from the default cell permissions on objects created in the directory by foreign users. For example, if the Initial Container Creation ACL of the directory **/.../abc.com/fs/usr/srivas** in the previous example had included the entry **foreign_other /.../abc.com**, the Object ACL of the **andi_files** subdirectory would have inherited the entry unchanged from the Initial Container Creation ACL. The entry would have granted users from the cell **abc.com** permissions on the subdirectory **andi_files**.

Because **andi**'s local cell is different from the default cell of the Initial Object Creation ACL and Initial Container Creation ACL of the parent directory of the **andi_files** subdirectory, entries on the corresponding ACLs that the subdirectory inherits are also changed as necessary. The new subdirectory inherits the following Initial Object Creation ACL and Initial Container Creation ACL from its parent:

```
dcecp> acl show /.:/fs/usr/srivas/andi_files -io
{mask_obj rw----}
{user_obj rw-c--}
{user andi rw----}
{foreign_user /.../abc.com/pierette rw----}
{foreign_user /.../ghi.com/pervaze r-----}
{group_obj r-----}
{other_obj r-----}
{foreign_other /.../def.com r-----}
dcecp> acl show /.:/fs/usr/srivas/andi_files -ic
{mask_obj rwx-id}
{user_obj rwxcid}
{user andi rwx-id}
{foreign_user /.../abc.com/pierette rwx-id}
{foreign_user /.../ghi.com/pervaze r-x---}
{group_obj r-x---}
{other_obj r-x---}
{foreign_other /.../def.com r-x---}
```

Mode Bits for New Objects That Do Not Inherit ACLs

The ACL inheritance algorithm described in "ACL Inheritance for Objects Created by Local Users" on page 78 applies only if the directory in which a file or directory object is created has the appropriate Initial Creation ACL. If the parent directory of a new object does not have the appropriate Initial Creation ACL, the object does not inherit an Object ACL; it initially has no Object ACL and is protected only with UNIX mode bits. DCE LFS bases the object's initial mode bits on the intersection of the following information:

- The UNIX mode bits specified at the system call level (with the UNIX **open( )**, **creat( )**, or **mkdir( )** system call) when the object is created. The application that invokes one of these system calls specifies the mode bits for the new object. For instance, when the UNIX **touch** command is used to create an object, the command usually specifies the **user**, **group**, and **other** mode bits as **r** and **w** in the resulting **creat( )** system call.
- The value of the UNIX file creation mask of the process that creates the object. The file creation mask filters the mode bits initially assigned to an object; the mask is defined with the UNIX **umask** command to be the octal complement of

the allowable mode bits. For instance, when a user creates an object, the value of the file creation mask of the user's process filters the mode bits assigned to the object.

For a new object, **user**, **group**, or **other** receives a mode bit only if the bit is specified in the system call *and* the bit is not filtered by the file creation mask. For instance, **user** for a new object receives read access only if the system call that creates the object specifies the **r** mode bit for **user** *and* the file creation mask of the creating process does not restrict **user** from having the **r** mode bit.

For example, the system call to create a new file typically specifies read and write access for **user**, **group**, and **other**; the file creation mask commonly restricts **group** and **other** to only read and execute access. When the file is created, **user** has the **r** and **w** bits, while **group** and **other** have only the **r** bit.

Similarly, the system call to create a new directory usually specifies read, write, and execute access for **user**, **group**, and **other**; the file creation mask commonly restricts **group** and **other** to only read and execute access. When the directory is created, **user** has the **r**, **w**, and **x** bits, while **group** and **other** have just the **r** and **x** bits.

A new directory whose parent directory has no Initial Container Creation ACL is created without an Initial Container Creation ACL. Likewise, a new directory whose parent directory has no Initial Object Creation ACL is created without an Initial Object Creation ACL.

Displaying Implicit (Nonexistent) ACLs: If you use the **dcecp acl show** command to display a nonexistent Object ACL, DCE LFS displays an implicit Object ACL. Although an Object ACL does not physically exist, DCE LFS constructs an implicit Object ACL whose entries and permissions match the object's UNIX mode bits.

For the implicit Object ACL of a directory, DCE LFS expands the **w** mode bit to grant the **i** and **d** ACL permissions in addition to the **w** permission. Until a directory has an Object ACL, DCE LFS must perform this expansion to allow for the creation of objects in the directory; without the **i** and **d** ACL permissions, the directory would effectively be read-only. Once the ACL exists, DCE LFS maps the **w** mode bit to just the **w** ACL permission, not the **i** and **d** permissions (see "ACL Interaction with UNIX Mode Bits" on page 74). Because the **i** and **d** ACL permissions are meaningless for a file, DCE LFS does not expand the **w** mode bit on the implicit Object ACL of a file.

If you use the **dcecp acl show** command to display a nonexistent Initial Creation ACL, DCE LFS displays an implicit Initial Creation ACL. It bases the permissions of this implicit ACL solely on a file creation mask of **0** (zero). In this case, DCE LFS ignores the file creation mask of the process that attempts to display the nonexistent Initial Creation ACL. For the implicit Initial Container Creation ACL, DCE LFS expands the **w** permission to grant the **i** and **d** permissions as well.

Like the **user_obj** entries of explicit ACLs (which physically exist), the **user_obj** entries of implicit ACLs grant the **c** permission. This permission ensures that the owner of an object can always change the ACLs of the object. Cell administrators for the File Server machine on which an object physically resides can also always change the ACLs of the object.

If you use the **dcecp acl show -ic** command to display the Initial Container Creation ACL of a directory that does not have this ACL, the command always displays an implicit Initial Container Creation ACL that has the following entries and permissions:

```
{user_obj rwxcid}
{group_obj rwx-id}
{other_obj rwx-id}
```

If you use the **dcecp acl show -io** command to display the Initial Object Creation ACL of a directory that does not have this ACL, the command always displays an implicit Initial Object Creation ACL that has the following entries and permissions:

```
{user_obj rwxc--}
{group_obj rwx---}
{other_obj rwx---}
```

Note again that, for an object created in a DCE LFS directory that does not have the appropriate Initial Creation ACL, DCE LFS considers the value of the file creation mask of the process that creates the new object when determining the object's initial mode bits. DCE LFS ignores the file creation mask of the calling process only when displaying nonexistent Initial Creation ACLs.

Creating Explicit (Existing) ACLs: The Object ACL and Initial Creation ACLs of an object that is protected only with UNIX mode bits remain implicit until you use the **dcecp acl** command to save them, at which point DCE LFS creates explicit ACLs for the object. For example, if you use the **dcecp acl modify** command to change an implicit ACL, DCE LFS creates an explicit ACL when it saves the changes.

Unless you change the permissions with the **dcecp acl** command that saves the ACL, the permissions granted by an Object ACL created with the command match the object's initial mode bits. Similarly, unless you change them with the **dcecp acl** command that saves the ACL, the permissions granted by an Initial Creation ACL created with the command are based on a file creation mask of **0** (zero).

Note that an Object ACL is always created for a file or directory created by a foreign user, even if the parent directory does not have the appropriate Initial Creation ACL. Because an unauthenticated user is treated as a user from an unknown foreign cell, an Object ACL is always created for an object created by an unauthenticated user also. The permissions granted by an Object ACL created in this way are based on the UNIX mode bits specified at the system call level when the object is created and the value of the UNIX file creation mask of the creating process.

Initial ACLs of a New Fileset

The root directory of a newly created DCE LFS fileset has no DCE ACLs. The directory is protected only with UNIX mode bits. Files and subdirectories created in the directory inherit UNIX mode bits according to the usual file system semantics. The root directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL remain implicit until the **dcecp acl** command is used to create explicit ACLs for the directory. (See"Creating Explicit (Existing) ACLs".)

A DCE LFS fileset can include many files and directories that never have ACLs. However, this approach fails to take advantage of the enhanced security available with DCE ACLs. Therefore, it is important to use the **dcecp acl** command to create the Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL for the root directory of a fileset *before* other objects are created in the directory.

For the root directory of a new DCE LFS fileset, **group** and **other** receive no permissions; **user** receives the UNIX **r**, **w**, and **x** mode bits. If the **dcecp acl show** command is used to view the directory's Object ACL, DCE LFS displays an implicit Object ACL that has the following entries and permissions:

```
{user_obj rwxcid}
{group_obj ------
{other_obj ------
```

If the **dcecp acl show** command is invoked with the **-io** or **-ic** option to view the Initial Container Creation ACL or Initial Object Creation ACL of a new root directory, DCE LFS displays an implicit Initial Creation ACL. DCE LFS constructs implicit Initial Creation ACLs for a new root directory just as it constructs implicit Initial Creation ACLs for any directory that does not have these ACLs. (See "Displaying Implicit (Nonexistent) ACLs" on page 84.)

Suggested Initial ACLs for a New Fileset

Cell administrators need to use the **dcecp acl** command to create the ACLs for the root directory of a new DCE LFS fileset. They should also manipulate the root directory and its ACLs to assign the directory the proper owner and give its ACL entries the appropriate permissions.

The owner of a fileset's root directory is initially set to **root**. A cell administrator must use the UNIX **chown** command or its equivalent to make the user who is to own the fileset the owner of the directory, thus granting that individual the **c** permission associated with the **user_obj** entry. A cell administrator should also use the UNIX **chgrp** command or its equivalent to change the owning group, as required.

Cell administrators may want to establish the convention of explicitly granting the owner of a new fileset all permissions on the fileset's root directory. In addition, they may want to limit the permissions initially granted by the **group_obj** and **other_obj** entries, changing these entries to grant only the **r** and **x** permissions. This practice allows all users from the local cell to list the contents of the directory and view the ACLs of the objects it contains, but little else.

The following example ACL provides the owner (**pierette**) of the root directory of a new fileset all permissions, granting all other users from the local cell just the **r** and **x** permissions:

```
dcecp> acl show /.../abc.com/fs/usr/pierette
{user_obj rwxcid}
{group_obj r-x---}
{other_obj r-x---}
```

Cell administrators should also apply these suggestions to the root directory's Initial Object Creation and Initial Container Creation ACLs. Because they are meaningless with respect to files, the **i** and **d** permissions do not need to be granted to the **user_obj** entry on the directory's Initial Object Creation ACL.

Recall that a user must have the **x** permission on each directory that leads to an object to access the object. Therefore, cell administrators should grant the **x** permission to the **group_obj** and **other_obj** entries on all directories that lead to common binary files. They should also grant the **x** permission to these entries on all directories that lead to the root directories of user filesets.

Delegation with DCE LFS Objects

Note: The information in this section applies only to applications that are written to use delegation.

The previous sections of this chapter document how ACLs work when you request access to an object directly. The information applies for most applications and for most routine file system operations. However, some applications may perform operations on an object on your behalf. An operation performed by such an application is referred to as a *delegation operation*; you delegate the operation to the server principal of the application.

For any operation, the user who initially requests the operation is referred to as the *initiator*. For a delegation operation, the principal that performs the operation for the initiator is known as the *delegate*. Because users typically delegate operations to server principals, delegation is usually described with respect to principals rather than users.

For an operation that does not involve delegation, only the initiator needs to have the permissions necessary to perform the requested operation. For a delegation operation, both the initiator and the delegate must have the permissions necessary to perform the operation. For example, suppose you request an application that executes as a delegate to print a file. In this case, you are the initiator because you have requested that the file be printed; the application that prints the file is the delegate because you have asked the application to print the file. Both you and the application need the permissions required to print the file.

With DCE ACLs, you can grant permissions to a principal that apply only when the principal is acting as a delegate on behalf of another principal. In the previous example, you could grant the application the necessary permissions for the requested file directly, or you could grant the application permissions only when it is acting as a delegate. Granting the application permissions directly allows the application to print the file on its own initiative, which can allow unauthorized users to print the file via the application. Granting the application permissions as a delegate ensures that the application prints the file only on behalf of authorized users.

Multiple delegates can be associated with a single operation. In this case, the collection of delegates is referred to as a *delegation chain*. The initiator and all delegates in the chain must have the permissions necessary to perform a requested operation. If the application in the previous example had forwarded your print request to a print server, both the application and the print server would have been members of the delegation chain for your print request. In this case, the initiator (you) and both delegates (the application and the print server) would have needed the permissions required to print the file.

The initiator of an operation is granted permissions via one of the standard entries described in Table 4 on page 64. However, delegates can also be granted permissions via special ACL entry types that exist exclusively for delegation. The following subsections provide more information about the additional ACL entries used for delegation and about how delegation works with DCE LFS objects.

ACL Entry Types for Delegation

For each ACL entry described in Table 4 on page 64, a corresponding entry of type **_delegate** is available. The following table describes the ACL entry types that can effectively be used for delegation with DCE LFS objects. (DCE LFS always ignores **user_obj_delegate**, **group_obj_delegate**, and **other_obj_delegate** entries, so these entries are omitted from the table.)

Table 6. ACL Entry Types for Delegation

Delegation Type	Key	Applies to
`user_delegate`	*principal_name*	The *principal_name* principal from the default cell acting as a delegate.
`foreign_user_delegate`	*cell_name*/ *principal_name*	The *principal_name* principal from the foreign cell *cell_name* acting as a delegate.
`group_delegate`	*group_name*	Members of the group *group_name* from the default cell acting as delegates.
`foreign_group_delegate`	*cell_name*/ *group_name*	Members of the group *group_name* from the foreign cell *cell_name* acting as delegates.
`foreign_other_delegate`	*cell_name*	Principals from the foreign cell *cell_name* who do not match any of the preceding entries acting as delegates.
`any_other_delegate`	None	Principals from any foreign cell who do not match any of the preceding entries acting as delegates.

Some examples of ACL entries for delegation follow:

{user_delegate print-server *permissions***}**
: Defines the permissions for the principal **print-server** from the default cell when the principal is acting as a delegate.

{group_delegate printers *permissions***}**
: Defines the permissions for members of the group **printers** from the default cell when members of the group are acting as delegates.

Each delegation entry can grant any of the permissions available for DCE LFS objects (**r**, **w**, **x**, **c**, **i**, and **d**). Each permission has the same meaning for a delegation entry that it has for a nondelegation entry. (See “ACL Permissions” on page 68 .)

ACL Evaluation for Delegation

DCE LFS performs an operation requested by one or more delegates only if the initiator and all delegates have the permissions required to perform the operation. If the initiator or one of the delegates does not have the required permissions, DCE LFS refuses to perform the operation. For example, to create a file in a directory, the initiator and all delegates must have the **w**, **x**, and **i** permissions on the directory; if the initiator or one of the delegates does not have all three of these permissions, the operation fails.

To determine the permissions for the initiator of an operation, DCE LFS considers only nondelegation ACL entries according to the evaluation algorithm presented in “ACL Evaluation” on page 70. However, to determine the permissions for a delegate, DCE LFS follows an evaluation algorithm that includes both delegation and nondelegation ACL entries. As with the usual evaluation algorithm, DCE LFS stops checking entries once a delegate meets a condition in the checking sequence; evaluation proceeds to a condition in the checking sequence only if the delegate fails to match all previous conditions.

The following list describes the order in which DCE LFS examines the entries on an ACL to determine the permissions for a delegate:

1. The delegate owns the object. DCE LFS grants the delegate the permissions from the **user_obj** entry.
2. A **user**, **user_delegate**, **foreign_user**, or **foreign_user_delegate** entry exists for the delegate. DCE LFS grants the delegate the permissions from the first of these entries that the delegate matches.
3. The delegate belongs to the group that owns the object (which acquires permissions via the **group_obj** entry) or to a group for which one of the following entries exists: **group**, **group_delegate**, **foreign_group**, or **foreign_group_delegate**. DCE LFS grants the delegate the permissions from all of the entries that the delegate matches.
4. The delegate is from the default cell. DCE LFS grants the delegate the permissions from the **other_obj** entry.
5. The delegate is from a foreign cell for which a **foreign_other** or **foreign_other_delegate** entry exists. DCE LFS grants the delegate the permissions from the first of these entries that the delegate matches.
6. The delegate is from a foreign cell and an **any_other** or **any_other_delegate** entry exists. DCE LFS grants the delegate the permissions from the first of these entries that exists.
7. The delegate matches no entry. DCE LFS denies the delegate access to the object.

Note that all delegation entries are always optional. Note also that a principal acquires permissions from a delegation entry only when acting as a delegate. A principal that is initiating an operation cannot obtain permissions from a delegation entry. Finally, DCE LFS filters all permissions granted via delegation entries through the **mask_obj** entry.

DFS Notes and Restrictions for Delegation

In most respects, delegation works with DCE LFS objects in the same way that it works with other DCE objects. You use **dcecp acl** commands to add, delete, modify, and display delegation entries. You can include delegation entries on Object ACLs and Initial Creation ACLs.

An object created through a delegation operation is owned by the last delegate in the delegation chain. For example, if you direct an application to create a file, and that application in turn directs another server process to create the file, the resulting file is owned by the server process that actually creates it, not by you or the application. In this case, you may not have the necessary permissions to perform further operations on the file.

Similarly, because non-LFS objects do not have DCE ACLs, the permissions required for a delegation operation that involves a non-LFS object are based solely

on the identity and permissions of the last delegate in the chain. The last delegate must acquire the necessary permissions by way of the **user**, **group**, or **other** mode bits.

The new ACL entry types introduced for delegation are incompatible with many programs from previous versions of DCE. Therefore, the following restrictions apply to the use of delegation entries on the ACLs of DCE LFS objects:

- Delegation is first available with Version 1.1 of DCE. In earlier versions of DCE, the **acl_edit** program was used to list and modify ACLs. Versions of the **acl_edit** program provided with versions of DCE earlier than 1.1 cannot display or modify ACLs that include delegation entries.
- File Server machines based on versions of DCE earlier than Version 1.1 cannot house filesets in which the ACLs of one or more objects include delegation entries. If the ACLs of one or more objects in a fileset include delegation entries, DFS does not allow you to do the following:
  - Move the fileset to a File Server that uses a version of DCE earlier than Version 1.1.
  - Add a replication site for the fileset on a File Server that uses a version of DCE earlier than Version 1.1.
  - Restore the fileset to a File Server that uses a version of DCE earlier than Version 1.1.

Finally, to use the identity of a chain of delegates for a delegation operation that involves DFS, an application sets the current login context to be that of the delegation chain for the operation (see the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for information about login contexts). When the operation is complete, the application sets the current login context back to its original state. This behavior is required only for delegation operations that involve requests to the File Exporter, which runs in the kernel; for operations that involve requests to user-space processes, applications can simply indicate that the login context of the delegation chain is to be used for the operation. (An application uses the **sec_login_set_context( )** routine to set the current login context; see the *IBM DCE for AIX, Version 2.2: Application Development Guide—Core Components* for more information.)

Note: DFS server processes that use administrative lists do not consider delegation when determining administrative privileges. The last delegate in the chain must be included in the appropriate administrative list to perform a privileged operation. For example, a privileged **bos** command requires that the last delegate in the chain be a member of the **admin.bos** list on the specified server machine. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for information about administering DFS server processes.)

Using Groups with DFS

Information about groups is maintained in the Registry Database by the DCE Security Service. (See the DCE Security Service portion of the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for complete details about creating and maintaining groups.)

Using groups allows you to assign permissions to several users at one time, rather than assigning them individually. You can create user groups or special interest

groups (for example, a group of all of the people from one department or a group of people who are working on one project) and then assign that group access to the appropriate files and directories.

You can also use groups to specify individuals who are permitted to perform administrative tasks; these individuals are specified in DFS administrative lists. DFS uses administrative lists to determine who is authorized to issue commands that affect filesets and server processes. Through administrative lists, you can precisely control the security in the administrative domains in your cell. This chapter does not discuss the management of administrative lists in detail. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for details about creating and maintaining administrative lists.)

You can also specify a group as an argument with certain DFS commands. The groups specified with these commands, like those included in certain administrative lists, define users who are allowed to issue commands that affect filesets. These groups are described in the following subsections.

Creating and Maintaining Groups

To authenticate to DCE, users must have accounts in the Registry Database (although some parts of DCE allow unauthenticated use). Part of the information associated with a user’s account is the user’s principal name and the groups and organizations to which the user belongs. Accounts are created and maintained by system administrators in the Registry Database, which is organized into three main directories: a person directory, a group directory, and an organization directory. (Some server machines run as separate authenticated principals; these servers also have accounts in the Registry Database. In the following section, the term *principal* refers to either a human user or a server machine.)

The collection of groups to which a user belongs is called a project list. A user acquires the access permissions granted to each group on the user’s project list. To assign a user to a group, use the **dcecp group add** command to add the user’s principal name to the group’s membership list in the Registry Database. (See the *IBM DCE for AIX, Version 2.2: Administration Commands Reference* for information about the **dcecp group** command.)

Using Groups with ACLs, Administrative Lists, and Commands

Groups can be used with ACLs, administrative lists, and certain DFS commands. Using groups in each of these ways provides a convenient way to specify several individuals with one entry.

ACLs specify access permissions for the users and groups that can perform operations on files and directories. Rather than specify an ACL entry for each member of a project on all project files, you can set up a group in the Registry Database that includes all project members. You can then specify the group on the files’ ACLs to provide all members the same access to the files.

Similarly, administrative lists specify the users and groups that can perform actions affecting specific server processes. Groups can be specified on the administrative list associated with each DFS server process. Often the same users need to be included on several administrative lists; these users can be specified as a group in the Registry Database and subsequently added to and removed from administrative lists as a group. For example, you can use a group to specify a system

administration team whose members need access to most DFS servers. Then, rather than modify all the administrative lists when the team membership changes, you can use the **dcecp group** command to modify the group in the Registry Database.

Groups can also be specified with options on certain DFS commands, including the **fxd** and **fts crserverentry** commands, to specify administrative users. Groups specified on the command lines of these commands differ from those specified with ACLs and administrative lists because only one group can be specified with these commands, but multiple groups can be specified with ACLs and administrative lists.

Suggestions for Administrative Groups

Administrative lists determine which users are permitted to perform privileged operations, such as restoring user files from backup copies or moving filesets from one server machine to another. Because they are stored on the local disk of each machine, administrative lists provide local control over a machine.

Each type of server process is associated with an administrative list, which allows you to differentiate between users who perform different administrative tasks. For example, administrative users who start and stop server processes need to be included on different administrative lists from users who manipulate filesets. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for details about the administrative tasks associated with each administrative list.)

Rather than specifying individuals in administrative lists, you can use groups in much the same way that you can use groups in ACLs. (You may want to use the same groups for ACLs and administrative lists in certain instances.) For example, you can create a large group of users for performing backup operations and include them on the administrative lists required to use the DFS Backup System (**admin.bak**, **admin.fl**, and **admin.ft**). A subset of this group can be included in the administrative list (**admin.bos**) for the BOS Server process on each machine in a domain, since that list designates the users and groups permitted to control server processes.

In two important cases, administrative users are specified as a group in command options. These groups are defined in the Registry Database, as are groups specified with ACLs and administrative lists; however, only one group can be specified with each of these commands.

The first command, **fxd**, initializes the File Exporter and starts related kernel daemons. The group specified with the command’s **-admingroup** option can change the ACLs and UNIX permissions associated with all file system objects exported from the File Server machine on which the File Exporter is running. Members of the group have the equivalent of the ACL **c** permission on all of the files and directories in each exported DCE LFS fileset, and they can effectively change the UNIX permissions on all files and directories in each exported non-LFS fileset. They can also change the owner and owning group of any file system object exported from the machine, and they can change the default cell of any DCE LFS object exported from the machine. Because they have access to all of the exported DCE LFS and non-LFS filesets on the File Server machine, members of this group should be both few in number and highly trusted.

Although inclusion in this administrative group is similar in many respects to being logged in as **root**, the two are not equivalent. A user who is logged into the local

machine as **root** can perform different operations on a file or directory, depending on whether the user accesses the object via its DCE pathname or via its local pathname.

The first way a user can access a file or directory is via the object's DCE pathname. For DCE access, DFS treats a user who is logged into the local machine as **root** but is not authenticated to DCE as the ***/.../*** *cellname***/hosts/***hostname***/self** principal of the local machine; in this case, the **root** user receives the permissions associated with the machine's **self** principal, which is treated as an authenticated user from the local cell. If the user is also authenticated to DCE as **root**, DFS treats the user according to the DCE identity **root**. The DCE identity **root** effectively has **root** privileges for data in all exported filesets in the cell, which is a serious security risk. Use the DCE **root** identity very cautiously or disable it altogether.

The second way a user can access a file or directory is via the object's local pathname. For local access, the **root** user has all of the privileges commonly associated with **root**; the **root** user can perform any file system operation on a file or directory. Note that a file or directory in a non-LFS fileset can always be accessed via a local pathname because a non-LFS fileset must always be mounted locally, as a file system on its File Server machine; a file or directory in a DCE LFS fileset can be accessed via a local pathname only if its fileset is mounted locally.

In summary, being a member of the **fxd** administrative group allows you to perform any operation on a file or directory in an exported fileset, but you may have to change the file's or directory's protections first. Being logged into the local machine as **root** lets you perform any operation on a file or directory in a locally mounted fileset immediately, without first changing the protections. Being authenticated as DCE **root** lets you perform any operation on a file or directory in an exported fileset immediately.

The second command, **fts crserverentry**, creates a server entry in the FLDB for a specified File Server machine. The group specified with the command's **-owner** option can administer entries in the FLDB for all filesets on the File Server machine. If the same group is given ownership of the server entries for all of the File Server machines in a domain, members of that group can then manipulate the FLDB entries for all filesets in the domain. Specifying a group with the **fts crserverentry** command is an alternative to specifying the same group in the **admin.fl** list, which would allow members of the group to access FLDB entries for filesets on all machines in the cell.

The number and size of a cell's administrative groups depend upon the organization of the cell. For example, a cell with a simple organization—one with a single administrative domain—could have the following two basic administrative groups:

- A group for cell-wide file system and fileset administrators (**cell_fileset**)
- A group for all server principals in the cell (**cell_servers**)

It could also include a third administrative group (**cell_file_system**). The members of this group would be a highly trusted subset of the members of the **cell_fileset** group. The following table lists the groups associated with each administrative list when only two groups are used and the groups associated with each administrative list when three groups are used. It also describes the function of the groups included in each list.

Table 7. Suggested Groups for Administering a Single-Domain Cell

Administrative List	With Two Groups	With Three Groups	Function
admin.bos	cell_fileset	cell_file_system	Manages server processes on each server machine
admin.fl	cell_fileset	cell_fileset	Creates server and fileset entries in the Fileset Location Database on each Fileset Database machine
admin.ft	cell_fileset cell_servers	cell_fileset cell_servers	Manages filesets on each File Server machine; moves filesets between File Server machines
admin.bak	cell_fileset	cell_fileset	Modifies the Backup Database on each Backup Database machine
admin.up	cell_servers	cell_servers	Allows upclient processes to obtain files from upserver processes on server machines

If two groups are used, the **cell_fileset** group is specified with both the **-admingroup** option of the **fxd** command and the **-owner** option of the **fts crserverentry** command for each File Server machine. With this configuration, the same select group of administrators manages the entire file system and all of the filesets in the cell.

If the third group, **cell_file_system**, is used, it replaces the **cell_fileset** group on the **admin.bos** lists on all server machines in the cell to allow its members to control the server processes on the machines. It also replaces the **cell_fileset** group on the **-admingroup** option of each **fxd** command for the File Server machines in the cell to enable its members to modify the permissions of all exported filesets in the cell.

An additional use of the **-owner** option of the **fts crserverentry** command and the **-admingroup** option of the **fxd** command is to allow owners of local workstations to export data from their local disks to the global namespace. In this case, a group consisting of the owners of a local workstation is specified with these options when a server entry is created for the workstation and when the File Exporter is initialized on the machine. (See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for more information about creating server entries.)

Chapter 4. Using Administrative Lists and Keytab Files

Most DFS server processes have an associated administrative list that defines the principals (users and server machines) and groups that can execute commands that affect the process. Server processes on different machines can have different lists, or each process can use a copy of the same list. Different types of processes can also share the same administrative list.

The management of an administrative domain is often shared by groups of administrative users. Each group is granted the privileges needed to execute specific commands on specific machines. By developing different groups, you have the flexibility to allow only certain people to perform specific tasks and access specific files. This allows you to simplify the administration of your domains by adding users to and removing them from groups rather than altering the administrative lists themselves.

You can use the **dcecp group create** command to create administrative groups. You can then use the **bos addadmin** command to place the groups on administrative lists. (See “Chapter 3. Using ACLs and Groups” on page 63 for more information about groups.)

Each DFS server machine also has a keytab file. The file contains server encryption keys, at least one of which is also stored in the cell’s Registry Database. Keytab files are used to provide security between server machines and client machines. A server machine uses an encryption key from the keytab file to prove that it is a valid server to clients accessing data from it, as well as to other server machines from which it accesses data.

This chapter provides information about using and managing administrative lists and keytab files. Administrative lists, keytab files, and encryption keys are maintained with **bos** commands. (Note that commands from the DCE Security Service are also available to manipulate keytab files and keys.)

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the Reference part of this guide and reference for complete details about each command.)

- The **-server** *machine* option specifies the server machine on which the command is to execute. This option names the machine on which the administrative list or keytab file to be affected is stored. The BOS Server on this machine executes the command. This option can be used to specify a server machine in a foreign cell.

  To run a privileged **bos** command (a **bos** command that requires the issuer to have some level of administrative privilege) using a privileged identity, always specify the full DCE pathname of the machine (for example, */.../abc.com/hosts/fs1*).

  To run an unprivileged **bos** command, you can use any of the following to specify the machine:

  – The machine’s DCE pathname (for example, */.../abc.com/hosts/fs1*)

  – The machine’s host name (for example, **fs1.abc.com** or **fs1**)

– The machine's IP address (for example, **11.22.33.44**)

Note: If you specify the host name or IP address of the machine, the command executes using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option); unless DFS authorization checking is disabled on the specified machine, a privileged **bos** command issued in this manner fails. If you specify the machine's host name or IP address, the command displays the following message (using the **-noauth** option suppresses the message):

```
bos:  WARNING: short form for server used; no
authentication information will be sent to the bosserver
```

When working with administrative lists, modify only the administrative lists stored on the System Control machine for the domain. The Update Server can then be used to distribute the lists to other server machines in the domain. If **-server** is not the System Control machine, the list is not distributed to other server machines in the domain. In addition, changes made to the list can be lost if the list is later updated from the System Control machine.

- The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See "Disabling DFS Authorization Checking on a Server Machine" on page 101 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.
- The **-localauth** option directs the **bos** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, */.../abc.com/hosts/fs1/dfs-server*. Do not confuse a machine's DFS server principal with its unique **self** identity. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for information about DFS server principals.)

  Use this option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

The **-noauth** and **-localauth** options are always optional.

Using Administrative Lists

Because administrative lists exist on a per-process and per-machine basis, different groups of principals can have different sets of administrative privileges within a domain. It is often useful, however, to have the same group or user on several lists. For example, the same users will probably administer filesets and the Fileset Location Database, so they should be included on all of the lists necessary to perform operations related to such administration.

In some cases, it is also practical to include the same users on multiple lists. For example, individuals listed in the **admin.bos** list can issue all **bos** commands, including those to add members to other administrative lists. Therefore, principals added to the **admin.bos** list should also be granted administrative privileges on the other administrative lists.

To simplify the management of these lists, use the domain's System Control machine as the source of all administrative lists for the domain. The System Control machine runs the **upserver** process; the other server machines in the domain run the **upclient** process. The **upclient** process takes updates of the administrative lists from the **upserver** process. As a result, all of the machines in the domain share the administrative lists and, thus, share a common set of administrators. (See the Reference part of this guide and reference for more information on the **upserver** and **upclient** processes.)

Administrative Lists

Many tasks require that users, groups, and machines be added to one or more administrative lists. Summaries of the different DFS administrative lists and the types of tasks associated with each list follow:

- The **admin.fl** list is associated with the Fileset Location (FL) Server. It designates the users and groups permitted to create server entries and fileset entries in the Fileset Location Database (FLDB). Because the FLDB is usually replicated to several different machines in the cell, you need to ensure that the **admin.fl** lists on all machines that house the FLDB are identical; otherwise, an administrator may be able to execute a command from one machine but not from another. You also need to ensure that the abbreviated DFS server principals of all Fileset Database machines are included in the **admin.fl** list (they can be present as members of a group); otherwise, the synchronization site for the FLDB may not be able to propagate changes to the database to the secondary sites.
- The **admin.ft** list is associated with the Fileset Server. It designates the users and groups permitted to administer filesets on a machine. Because some fileset operations (such as moving filesets) affect multiple machines, the server principal names of the machines involved in the operations must also be in this administrative list. To simplify management, it is best that the server principal names of all server machines in the domain be represented in the **admin.ft** list on the System Control machine so that the list is distributed to all File Server machines in the domain. Note that the server principals can be included directly, or a group to which they belong can be included.
- The **admin.up** list is associated with the Update Server. It contains the server principals for all server machines in the domain, allowing the **upclient** processes on those machines to obtain files such as common configuration files, binary files, and administrative lists from the **upserver** process. The list should be stored on machines such as the System Control machine and the Binary Distribution machine, which run the **upserver** process.
- The **admin.bos** list is associated with the BOS Server. It designates the users and groups permitted to create, start, and stop DFS server processes and other processes to be controlled by the BOS Server on a machine. The BOS Server runs as **root**, so processes that it starts run with **root** privileges. Because they can direct the BOS Server to start any process, and because they can add and remove members from the other administrative lists on the machine, users in the **admin.bos** list are usually a subset of the users in the other lists for a machine or domain.
- The **admin.bak** list is associated with the Backup Server. It designates the users and groups allowed to issue commands in the **bak** command suite. These commands are used to configure the Backup System and to dump and restore data. The Backup Database, like the FLDB, is typically replicated to several different machines in the cell. Therefore, you need to ensure that the **admin.bak** lists on all machines that house the Backup Database are identical. You also need to ensure that the **admin.bak** list includes the abbreviated DFS server

principals of all Backup Database machines to make sure that the synchronization site for the Backup Database can propagate changes to the secondary sites (the server principals can be present as members of a group).

Many tasks require that a user be included on multiple lists; for example, to move a fileset from one server machine to another, you must be included in the **admin.ft** file on the source machine, and you and the server principal for the source machine must be listed in the **admin.ft** list on the destination machine. You must also be included in the **admin.fl** list on all machines on which the FLDB is stored. The check by the DFS server processes to ensure that the issuer of a command is included in the proper administrative lists is referred to as *DFS authorization checking*.

In this guide, the specific privileges required to execute commands are detailed with each task. (See the Reference part of this guide and reference for complete information about the administrative privileges and permissions required to issue each DFS command.) Note that the names of the administrative lists are only recommendations; different names can be specified when the respective processes are started.

Maintaining Administrative Lists

Administrative lists for server processes can initially be created in one of two ways:

- A server process automatically creates its administrative list when it is started on a machine if the list does not already exist on the local disk of the machine. By default, a process places its list in the configuration directory, *dcelocal*/**var/dfs**. An administrative list generated by a process is always empty.
- You can create an administrative list for any process except the BOS Server by including the **-createlist** option with the **bos addadmin** command. Because the BOS Server must be running to issue the **bos addadmin** command, and because every process creates its administrative list if the list does not already exist when the process starts, the **admin.bos** list *must* already exist when you issue the **bos addadmin** command.

Every server machine stores administrative lists for its processes on its local disk. It is recommended that all administrative lists be stored in the default directory, *dcelocal*/**var/dfs**. If the administrative list for a process is stored in a different directory, you must specify the full pathname of the list when you start the process. For example, if you store the **admin.bos** file in a directory called *dcelocal*/**var/dfs/config**, you must use that pathname when you start the **bosserver** process on that machine.

Do not create multiple copies of administrative lists and store them in different directories; this can cause confusion when attempting to determine who has administrative privilege and can potentially result in unauthorized users executing restricted commands. Note that a Private File Server machine typically has specialized versions of the **admin.bos** and **admin.ft** administrative lists to allow its administrators to manage its processes and the data it contains. Such lists can reside in the *dcelocal*/**var/dfs** directory, but they should not be retrieved from the System Control machine via the Update Server.

To guarantee that all users and groups have the same privileges on all server machines, the same users and groups must be on the administrative lists that grant those privileges on each machine. If the same copy of an administrative list is not distributed to all machines in the domain, users can be prohibited from issuing

commands on specific machines. For instance, suppose a user is listed in the **admin.ft** file on machine **fs1** but is not listed in the **admin.ft** file on machine **fs2**. The user can issue commands that affect filesets on **fs1**, but the user cannot issue commands that affect filesets on **fs2**.

To maintain consistency among administrative lists, use the following guidelines:

- Make all changes only to the files stored on the domain's System Control machine.
- Ensure that all other server machines in the domain are running the **upclient** process to reference the appropriate administrative lists on the System Control machine. The **upclient** and **upserver** processes then automatically maintain the synchronization of the administrative lists.

You can remove an administrative list that you no longer need by including the **-removelist** option with the **bos rmadmin** command. If you use the command to remove the last member from an administrative list or if a list contains no members when you issue the command, the **-removelist** option specifies that the list is to be removed. The option has no effect if the list is not empty.

Listing Principals and Groups in Administrative Lists

Issue the **bos lsadmin** command to check the principals and groups on an administrative list:

$ bos lsadmin -server *machine* **-adminlist** *filename*

The **-adminlist** *filename* option specifies the name of the administrative list to be displayed. The default directory for the administrative lists is the configuration directory, *dcelocal***/var/dfs**. If the lists are stored in the default directory, you need to provide only the specific filename, **admin.fl**, **admin.ft**, **admin.up**, **admin.bos**, or **admin.bak**. If the lists are stored elsewhere, you must enter the pathname that was used when the specific process was started.

For example, the following command lists the members of the **admin.bos** file on the server machine named **fs1**:

$ bos lsa /.../abc.com/hosts/fs1 admin.bos

```
Admin Users are: user: jones, user: smith,
user: hosts/fs1/self, group: dfs-admin, group: fs1-admin
```

Adding Principals and Groups to Administrative Lists

To add principals and groups to an administrative list, do the following:

- Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the administrative list to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list.
- Issue the **bos addadmin** command to add principals, groups, or both to an administrative list:

  $ bos addadmin -server *machine* **-adminlist** *filename* \
  $ [-principal *name***...] [-group** *name***...] [-createlist**]

  The **-adminlist** *filename* option specifies the name of the administrative list to which principals and groups are to be added. The default directory for the administrative lists is the configuration directory, *dcelocal***/var/dfs**. If the lists are

stored in the default directory, you need to provide only the specific filename, **admin.fl**, **admin.ft**, **admin.up**, **admin.bos**, or **admin.bak**. If the lists are stored elsewhere, you must enter the pathname that was used when the specific process was started.

The **-principal** *name* option specifies the principal name of each user or server machine to be added to the list. A user from the local cell can be specified by a full or abbreviated principal name (***/.../*** *cellname****/*** *username* or just *username*, for example); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** *cellname****/hosts/*** *hostname****/self*** or just **hosts/** *hostname****/self***); a server machine from a foreign cell can be specified only by a full principal name.

The **-group** *name* option specifies the name of each group to be added to the list. A group from the local cell can be specified by a full or abbreviated group name (for example, ***/.../*** *cellname****/*** *group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

The **-createlist** option specifies that the administrative list indicated with **-adminlist** is to be created if it does not already exist. Any principals or groups specified with the command are added to the new file; if no principals or groups are specified, the command creates an empty file. This option has no effect if the specified file already exists.

Removing Principals and Groups from Administrative Lists

To remove principals and groups from an administrative list, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the administrative list to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list.
2. Issue the **bos rmadmin** command to remove principals, groups, or both from an administrative list:

   ```
   $  bos rmadmin -server machine -adminlist filename \
   [-principal name...] [-group name ...] [-removelist ]
   ```

   The **-adminlist** *filename* option specifies the name of the administrative list from which to remove principals and groups. The default directory for the administrative lists is the configuration directory, *dcelocal****/var/dfs***. If the lists are stored in the default directory, you need to provide only the specific filename, **admin.fl**, **admin.ft**, **admin.up**, **admin.bos**, or **admin.bak**. If the lists are stored elsewhere, you must enter the pathname that was used when the specific process was started.

   The **-principal** *name* option specifies the principal name of each user or server machine to be removed from the list. A user from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** *cellname****/*** *username* or just *username*); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** *cellname****/hosts/*** *hostname****/self*** or just **hosts/** *hostname****/self***); a server machine from a foreign cell can be specified only by a full principal name.

   The **-group** *name* option specifies the name of each group to be removed from the list. A group from the local cell can be specified by a full or abbreviated

group name (***/.../*** *cellname****/*** *group_name* or just *group_name*, for example); a group from a foreign cell can be specified only by a full group name.

The **-removelist** option specifies that the administrative list indicated with **-adminlist** is to be removed if it is empty either when the command is issued or after any principals or groups specified with the command are removed. This option has no effect if the specified file is not empty when the command is issued or after any indicated principals or groups are removed.

Be sure to remove entries from the DFS administrative lists before removing the principal or group from the DCE Security registry. If you delete a DCE principal or group ID from the registry database without first deleting it from the DFS administrative lists, the entry will be displayed in UUID format by the **bos lsadmin**, since the UUID can no longer be translated to a valid DCE principal.

The **bos rmadmin** command will display the following error message:

```
bos: failed to delete user '000002f4-e9d9-2e48-a800-10005aa8b1d4',
(unexpected internal error; check bosserver log (dfs / bbs ))
```

Disabling DFS Authorization Checking on a Server Machine

DFS authorization checking involves a server process checking the proper administrative list to ensure that the issuer of a command has the necessary administrative privilege to execute the command. If the issuer is a member of the list, the process performs the requested operation; if the issuer is not a member of the list, the process does not perform the operation.

By default, DFS authorization checking is enabled on every server machine. You can disable it on a machine by

- Including the **-noauth** option with the **bosserver** command when the BOS Server is started on the machine.
- Issuing the **bos setauth** command and specifying the machine with the command's **-server** option.
- Manually creating the zero-length file */.../dcelocal****/var/dfs/NoAuth*** on the local disk of the machine; the first two methods create this file automatically.

All DFS server processes, including the BOS Server, check for the presence of the **NoAuth** file when they are requested to perform an operation. They do not check for the necessary administrative privilege for a requested operation when the file is present. Consider disabling authorization checking on a machine in the following situations:

- During initial DFS installation, by including the **-noauth** option with the **bosserver** command. Before administrative lists have been created or users have been added to the lists, no one has the necessary privilege to issue an administrative command.
- If some component of the Security Service is unavailable, by manually creating the **NoAuth** file. If the **secd** process or a related security process is unavailable, the issuer of a command cannot acquire the security credentials necessary to allow DFS server processes to verify administrative privilege. In this case, the **-noauth** option must be included with a command to bypass the unavailable Security Service. (See "Using the **-noauth** Option" on page 102.)

- During server encryption key emergencies, by manually creating the **NoAuth** file. Improper keys may make it impossible for DFS server processes to verify a user's administrative privilege. The **-noauth** option can again be used to circumvent the security problems.
- To view the actual keys stored in a keytab file, by issuing the **bos setauth** command. If authorization checking is enabled, checksums are displayed rather than the actual keys. (See "Using Keytab Files" on page 103.)

Never disable DFS authorization checking for longer than is absolutely necessary. Disabling DFS authorization checking on a machine compromises security by allowing anyone, including the unprivileged identity **nobody**, to execute any DFS command on the machine. To enable DFS authorization checking (the normal state) once it has been disabled, use the **bos setauth** command. Use the **bos status** command to determine whether DFS authorization checking is enabled or disabled on a server machine.

Using the -noauth Option

Most DFS commands from the **bos** and **fts** suites have an optional **-noauth** option. Omitting the **-noauth** option from a command requires that authentication information be available about the issuer of the command; because it is optional, the option is always omitted by default. Including the **-noauth** option with a command directs the **bos** or **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. Include the **-noauth** option with a command if

- Authentication information is unnecessary. If DFS authorization checking is disabled on a server machine or if a command does not require administrative privilege, DFS server processes on the machine do not check for authentication information. Omitting the **-noauth** option in these cases causes the **bos** or **fts** program to include the unnecessary security credentials of the issuer with the command; including the **-noauth** option allows the command to execute more quickly because it avoids the unnecessary creation of the issuer's security credentials.

  You may want to include the **-noauth** option with a command that does not require administrative privilege if the command is to be issued as a **bos cron** process. (See "Chapter 5. Monitoring and Controlling Server Processes" on page 111 for more information about **bos** processes.)
- Authentication information is unavailable. If some aspect of the Security Service is unavailable (for example, if the **secd** process is not functioning) and the **-noauth** option is omitted from a command, **bos** and **fts** commands fail even if DFS authorization checking is disabled. The failure occurs because the **bos** or **fts** program cannot obtain the issuer's security credentials from the Security Service. In such cases, even commands that do not require administrative privilege may fail if the **-noauth** option is not used.

Including the **-noauth** option when DFS authorization checking is disabled ensures that a command will succeed because the Security Service is never contacted to assemble the issuer's security credentials. Include the **-noauth** option with a command that requires administrative privilege only if DFS authorization checking is disabled on the necessary machines. A command that requires administrative privilege fails if the **-noauth** option is included and DFS authorization checking is not disabled.

Disabling or Enabling DFS Authorization Checking

To disable or enable DFS authorization checking, do the following:

CAUTION:

Disabling DFS authorization checking makes potentially serious security breaches possible. Enable DFS authorization checking as soon as the need to have it disabled has passed.

1. If DFS authorization checking is to be disabled, verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which checking is to be disabled. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list.
2. Enter the **bos setauth** command to disable or enable DFS authorization checking on the server machine:

$ bos setauth -server *machine* **-authchecking {on | off}**

The **-authchecking off** option disables DFS authorization checking by creating the **NoAuth** file on the machine specified with **-server** ; the **-authchecking on** option enables DFS authorization checking by removing the **NoAuth** file from the machine specified with **-server**.

Using Keytab Files

An encryption key is a set of octal characters used to encrypt and decrypt packets of information. In DFS, a server encryption key is employed to provide security for information transferred between server processes and their clients. An encryption key for a server is analogous to a password for a user. All DFS server processes on a server machine use the same key from the keytab file as a ″password″ for that machine.

One or more keys are stored in the **/krb5/v5srvtab** keytab file on the local disk of each server machine. Each key is associated with a principal name, usually the DFS principal name of the machine on which the key resides. Multiple keys can be associated with a principal name in a keytab file, but one key (usually the most recent) is also stored in the Registry Database for any principal name in a keytab file.

The key stored in the Registry Database is the one used for subsequent communications between processes on client machines and processes on the server machine. Multiple keys can exist if a new key is added while an existing key is still being used for communications between a client and server. Note that, once communications have been initiated between a client and server using a key, removing that key may not prevent continued communications between the two.

Maintaining Keytab Files

Maintaining server encryption keys and keytab files is critical to establishing adequate security measures in your cell or domain. Under normal circumstances, keytab files require little maintenance. Because they are analogous to user passwords, they should be changed about as often.

The first step in changing a server encryption key is to add a new key to the keytab file. Two commands are available for adding keys: **bos genkey** and **bos addkey**.

- The **bos genkey** command automatically generates a random key. It also automatically updates the entry in the Registry Database for the principal with which the key is associated. Any subsequent communications that involve the specified principal and that require a key use the newly added key.
- The **bos addkey** command performs a similar function, but it requires that you enter a string to be converted into a key, and it gives you the option of updating the Registry Database entry for the indicated principal. The **bos addkey** command is less secure than the **bos genkey** command because user-specified strings are seldom as random as machine-generated strings.

A keytab file must already exist before either of these commands can be used to add a key to it; keytab files are created with the **dcecp keytab create** command.

A unique version number is associated with each key for a principal in a keytab file. When adding a key to a keytab file, you must specify its key version number as one of the following:

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine. Because reusing a version number currently in use in a keytab file can cause authentication failures between the processes on a server machine and clients communicating with them, an error is returned if you attempt to do so.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database. However, it may not be unique for the indicated principal in the keytab file on the specified machine, in which case it replaces the key currently associated with the principal/version number pair in the keytab file.

It is best to keep the key version numbers in sequence by choosing a number that is one greater than the current version number for the principal. Use the **bos lskeys** command to examine the key version numbers associated with the keys in a keytab file.

The **bos lskeys** command also displays a **checksum** with each key version number. A checksum is a decimal number derived by encrypting a constant with a key. Because displaying the checksum is adequate for most purposes (for example, when checking key version numbers presently in use), and because its display is less of a security risk, it is displayed rather than the actual key associated with a version number. Note that the actual keys can be viewed by first issuing the **bos setauth** command to disable DFS authorization checking on the server machine; however, because disabling DFS authorization checking creates a compromised state of security, it is not recommended.

After a new key has been added to a keytab file, the old key can be removed from the file. The **bos rmkey** command can be used to remove one or more keys from a keytab file. Removing the key currently in use in the Registry Database or any other key still being used for client/server communications can cause authentication failures between server processes and clients. Tickets based on a removed key are invalidated; new tickets based on a new key must be obtained to reestablish communications with the server process.

To prevent authentication failures, wait until all old tickets held by client machines expire before removing the old key. For example, if tickets held by clients expire

after 2 hours, wait at least that long from the time the new key is added to remove the old key. If you are unsure of whether a key is still in use, use the **bos gckeys** command to delete, or ″garbage collect,″ those keys from a keytab file that are no longer in use (obsolete).

Note: The BOS Server uses authenticated RPC for communications with clients. By default, it uses the packet privacy protection level with the **bos** key commands described in this chapter. However, this protection level is not available to everyone who uses DCE. If it is not available to you, the BOS Server uses the next-highest protection level, packet integrity. It displays the following message, reporting that it must use the packet integrity protection level because packet privacy is not available:

```
Data encryption unsupported by RPC.  Continuing without it.
```

Listing Keys in Keytab Files

To list the keys in a keytab file, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine whose keys are to be displayed. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Issue the **bos lskeys** command to view the key version numbers and checksums from the keytab file on a server machine:

   $ bos lskeys -server *machine* **[-principal]** *name*

   The **-principal** *name* option is the principal name for which associated keys are to be listed. The default is the DFS principal name of the machine specified with **-server**.

Adding Keys to Keytab Files

To add a key to a keytab file, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be include in the **admin.bos** list on the machine on which the keytab file to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Verify that the DFS server principal of the machine whose keytab file is to be affected has the necessary permissions to alter entries in the Registry Database. (See the Security Service portion of the *IBM DCE for AIX, Version 2.2: Administration Guide—Core Components* for more information.)
3. Choose a key version number for the new key. If necessary, issue the **bos lskeys** command to check the version numbers of the keys in the appropriate machine's keytab file:

   $ **bos lskeys -server** *machine* **[-principal** *name*]

   The **-principal** *name* option is the principal name for which associated keys are to be listed. The default is the DFS principal name of the machine specified with **-server**.
4. Create a new key in the keytab file with either the **bos genkey** command or the **bos addkey** command. The **bos genkey** command is the more secure of the two commands. It generates a random, octal string for use as the key. It also automatically updates the Registry Database in addition to adding the key to the keytab file.

```
$  bos genkey -server machine -kvno +_or_version_number \
[-principal  name ]
```

The **-kvno** *+_or_version_number* option is the key version number of the new key. Valid arguments for this option are

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database, but it may not be unique for the indicated principal in the keytab file on the specified machine.

The **-principal** *name* option is the principal name with which the key is to be associated. The default is the DFS principal name of the machine specified with **-server**.

The **bos addkey** command is less secure because it requires you to enter a string to be converted into the key. However, you can include the **-localonly** option with the command to add the key to the keytab file without updating the Registry Database, which is useful for certain server encryption key emergencies.

```
$ bos addkey -server machine -kvno +_or_version_number \
-password string[-principal name] [-localonly ]
```

The **-kvno** *+_or_version_number* option is the key version number of the new key. Valid arguments for this option are

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database, but it may not be unique for the indicated principal in the keytab file on the specified machine.

The **-password** *string* option is a character string to be converted into an octal string. The string can include any characters, including spaces if it is enclosed in ″ ″ (double quotes).

The **-principal** *name* option is the principal name with which the new key is to be associated. The default is the DFS principal name of the machine specified with **-server**.

The **-localonly** option specifies that the key is to be added to the keytab file on the machine indicated by **-server**, but the Registry Database is not to be updated.

5. If you added the key to the keytab file by using the **bos addkey** command and its **-localonly** option, use the **dcecp keytab add** command with the **-registry** option to add the key to the Registry Database when necessary.

Removing Specific Keys from Keytab Files

To remove a specific key from a keytab file, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the keytab file to be

affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.

2. Remove one or more keys from the keytab file with the **bos rmkey** command:

   ```
   $ bos rmkey -server machine -kvno version_number... \
    [-principal  name]
   ```

   The **-kvno** *version_number* option is the key version number of each key to be removed for the indicated principal. Valid arguments for this option are integers in the range 1 to 255.

   The **-principal** *name* option is the principal name associated with the keys to be removed from the keytab file. The default is the DFS principal name of the machine specified with **-server** .

Removing All Obsolete Keys from Keytab Files

To remove all obsolete keys from a keytab file, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the keytab file to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Remove obsolete keys (those keys that are no longer in use) from the keytab file with the **bos gckeys** command:

   ```
   $ bos gckeys -server machine [-principal  name]
   ```

   The **-principal** *name* option is the principal name for which obsolete keys are to be removed from the keytab file. The default is the DFS principal name of the machine specified with **-server**.

Handling Server Encryption Key Emergencies

Server encryption key emergencies are situations that require immediate attention to ensure continued, authenticated communications between the processes on a server machine and the clients with which they are communicating. One type of emergency occurs when you suspect that a machine's encryption key in the Registry Database is compromised. In this case, you must immediately remove that key from the keytab file and reboot the server machine to prevent unwanted access to the server.

A second type of server encryption key emergency can result from the current key becoming corrupted. In this case, server processes using the key cannot decrypt the information used in client/server communications, bringing all activity involving those processes to a halt. You must remove the corrupted key from the keytab file, but you do not need to reboot the server machine. From a security perspective, this type of emergency is less severe than one resulting from a compromised key, but it requires immediate attention nonetheless.

To resolve encryption key emergencies, you must add a new server key to both the keytab file on the machine and the Registry Database. You must turn off DFS authorization checking when handling key emergencies. Because disabling DFS authorization checking is a severe security risk, disable authorization checking for a minimal amount of time.

The emergency procedure requires you to be logged into the affected server machine as **root** to create the **NoAuth** file and to reboot the machine. Many of the

steps in the procedure were detailed in previous sections of this chapter. (See “Chapter 5. Monitoring and Controlling Server Processes” on page 111 for a description of the **bos shutdown** command.)

Note: Rebooting is not necessary when replacing a corrupted key. It may not always be necessary when dealing with a compromised key; for example, it may be sufficient simply to restart any processes associated with the compromised key. However, rebooting the machine is the safest way to terminate all unauthorized communications.

1. Log in as **root** on the affected machine.
2. Disable DFS authorization checking by creating the *dcelocal***/var/dfs/NoAuth** file. It is usually recommended that you use the **bos setauth** command to create the **NoAuth** file. However, because the server encryption key emergency can make it impossible to issue **bos** commands, create the file with the **touch** command (or its equivalent).
3. Use the **bos lskeys** command to check the key version numbers currently in use, using the **-noauth** option to employ an unprivileged identity as the identity of the issuer of the command:

   # bos lskeys -server *machine* **[-principal** *name* **]-noauth**

   The **-principal** *name* option is the principal name for which associated keys are to be listed. The default is the DFS principal name of the machine specified with **-server**.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.
4. Create the new key with the **bos genkey** command, specifying a new key version number for the key with the **-kvno** option and again using the **-noauth** option:

   ```
   # bos genkey -server machine -kvno +_or_version_number \
    [-principal  name ] -noauth
   ```

   The **-kvno** *+_or_version_number* option is the key version number of the new key. Valid arguments for this option are

   - An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine.
   - **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database, but it may not be unique for the indicated principal in the keytab file on the specified machine.

   The **-principal** *name* option is the principal name with which the key is to be associated. The default is the DFS principal name of the machine specified with **-server**.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.
5. Use the **bos rmkey** command to remove any old keys that are compromised. Specify the version number of each key to be removed with the **-kvno** option, and again use the **-noauth** option.

   ```
   # bos rmkey -server machine -kvno version_number...\
   [-principal name]-noauth
   ```

The **-kvno** *version_number* option is the key version number of each key to be removed for the indicated principal. Valid arguments for this option are integers in the range 1 to 255.

The **-principal** *name* option is the principal name associated with the keys to be removed from the keytab file. The default is the DFS principal name of the machine specified with **-server**.

The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

6. *If the emergency resulted from a compromised key*, issue the **bos shutdown** command to prepare to reboot the machine. You must reboot the machine to terminate all existing communications that are based on the compromised encryption key. The **bos shutdown** command directs the BOS Server to shut down the other DFS server processes running on the machine. Include the **-wait** option with the command to be sure that all processes have stopped before continuing.

 # bos shutdown -server *machine* **-wait -noauth**

The **-wait** option delays the command shell prompt's return until the processes are stopped. If the option is omitted, the prompt returns immediately, even if the processes are not yet stopped.

The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

7. Enable DFS authorization checking by entering the **bos setauth** command. Specify the value **on** with the **-authchecking** option, and include the **-noauth** option.

 # bos setauth -server *machine* **-authchecking {on | off} -noauth**

The **-authchecking on** option enables DFS authorization checking by removing the **NoAuth** file from the machine specified with **-server** ; **-authchecking off** disables authorization checking by creating the **NoAuth** file on the machine specified with **-server**.

The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

8. *If the emergency resulted from a compromised key*, issue the appropriate reboot command (**/etc/reboot** or its equivalent) for the machine to be rebooted. For example:

```
# /etc/reboot
```

Note: On AIX, use the **/usr/sbin/shutdown** command.

The dcecp keytab Command and Keytab Files

The **dcecp keytab** command can also be used to manipulate encryption keys in keytab files. The following **dcecp keytab** operations are used to manage keytab files:

add Adds keys to a keytab file and, optionally, to the Registry Database

catalog
Lists the names of all keytab files on a machine

create Creates a keytab file

delete Deletes an entire keytab file or, optionally, just the keys in a keytab file

remove
Removes keys from a keytab file

show Lists the keys in a keytab file

These operations perform functions similar to those of their counterparts in the **bos** command suite. Whereas the analogous **bos** commands require that you be included in the **admin.bos** list on the machine whose keytab file you wish to manage, these operations require that you have the necessary ACL permissions. (See the *IBM DCE for AIX, Version 2.2: Administration Commands Reference* for more information about the **dcecp keytab** command.)

Chapter 5. Monitoring and Controlling Server Processes

To provide efficient and correct operation, the processes that are running on DFS server machines in a cell must be configured properly. The Basic OverSeer Server (BOS Server) continually monitors and, if necessary, restarts the other server processes on a machine; you specify the processes that the BOS Server is to monitor. The BOS Server runs on all DFS server machines.

You also control server process status by issuing **bos** commands to perform routine maintenance or to correct errors the BOS Server cannot correct by itself. This chapter explains how to define a server machine's processes and how to start and stop them. The BOS Server can monitor and control processes other than DFS processes. However, the information in this chapter refers specifically to DFS server processes.

Do not use the BOS Server to control the following processes on a machine: **fxd**, **dfsd**, **dfsbind**, or **dfsexport**. The first two processes spawn kernel threads that, if continually restarted, can eventually result in system failure on the machine. The last two processes are usually executed only when a machine initially starts, and they must be started in the proper sequence with respect to other processes. It is recommended that all four of these processes be started by including a line in the proper initialization file (**/etc/rc** or its equivalent).

Process Entries in the BosConfig File

You define which processes the BOS Server monitors by creating process entries in the *dcelocal***/var/dfs/BosConfig** file on the local disk of each server machine. The information in a process entry defines how the process is to run. You control the process status (**Run** or **NotRun**) by changing the entry with **bos** commands. When the BOS Server starts, it creates a **BosConfig** file with no process entries if the file does not already exist.

The order in which process entries are added to or appear in the **BosConfig** file is irrelevant. The BOS Server restarts multiple processes virtually simultaneously. However, do not depend on one process starting before another simply because its entry precedes that of the other process in the **BosConfig** file. The BOS Server has no control over how long a process takes to start.

CAUTION:
Do not directly edit the information in the BosConfig file; use only the commands described in this chapter to alter the file. Directly editing the BosConfig file can result in changes to process entries of which the BOS Server is unaware. Such changes do not take effect until the BOS Server is restarted and again reads the file.

Each process entry includes the following information about its process:

- Its name. The name that appears in the **BosConfig** file for a process is the name used to refer to that process with any **bos** commands that require a process name.
- Its type. The type can be one of the following:
  - **simple**: A **simple** process is a continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes. This process has a single parameter: the command to be executed.

  - **cron**: A **cron** process, like a **simple** process, runs independently of other processes; however, a **cron** process runs periodically, not continuously. This process has two parameters: the first is the command that is to be executed; the second is the time that the command is to be executed.

- Its status flag. Status flags are for internal use only and do not appear in any output. The flag can have one of the following values:
  - **Run**, meaning the process needs to run whenever possible. The BOS Server starts the process initially at reboot and restarts it automatically if it fails at any time. This flag is used to keep a process running at all times; for example, to ensure that the **ftserver** process on a File Server machine runs continuously. (The **Run** status flag appears in the **BosConfig** file as a **1**.)
  - **NotRun**, meaning the process never runs. The BOS Server never starts or automatically restarts the process; the process runs only when you instruct the BOS Server to start it. This flag is used to stop a process for an extended period of time; for example, to stop the **upclient** process from accessing new binary files while you test the current binaries. (The **NotRun** status flag appears in the **BosConfig** file as a **0**.)
- Its command parameters. These parameters are used by the BOS Server to run the process.

The following output from the **bos status** command displays an entry from the **BosConfig** file. (See “Listing Status and Machine Information” on page 117 for more information about the **bos status** command, which is used to list entries from the **BosConfig** file.)

```
Instance ftserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:02 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/ftserver'
```

It is possible for the BOS Server’s memory state to change independently of the **BosConfig** file. The BOS Server checks the **BosConfig** file whenever it starts or restarts, (in response to the **bos restart** command, at the general restart time, or at system reboot). At that time, the BOS Server transfers information from the file into memory and does not read the file again until it restarts.

Therefore, it is possible to use the **bos shutdown** command to stop a running process, even though its status flag in the **BosConfig** file is **Run**. Similarly, you can use the **bos startup** command to start a process running by setting its memory state status flag to **Run** without setting its status flag in the file to **Run**. The commands discussed in this chapter can affect the BOS Server’s memory state, the information in the **BosConfig** file, or both.

Starting or stopping certain processes, either temporarily or permanently, has an effect on the other processes that run on the other server machines in your cell. For example, an **upserver** process must run on each System Control machine and Binary Distribution machine. If you start or stop the process on one machine, you must start it on a replacement System Control or Binary Distribution machine. You must also modify the **upclient** processes on the appropriate server machines so that they reference the new System Control or Binary Distribution machine.

Standard Information in this Chapter

The following subsections present options and arguments common to many of the commands described in this chapter. It also presents some common operations that are explained in the chapter or that can be useful when performing other operations.

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the Reference part of this guide and reference for complete details about each command.)

- The **-server** *machine* option specifies the server machine on which the command is to execute. This option names the machine whose process or file is to be affected. The BOS Server on this machine executes the command. This option can be used to specify a server machine in a foreign cell.

  To run a privileged **bos** command (a **bos** command that requires the issuer to have some level of administrative privilege) using a privileged identity, always specify the full DCE pathname of the machine (for example, */.../abc.com/hosts/fs1*).

  To run an unprivileged **bos** command, you can use any of the following to specify the machine:

  – The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*)
  – The machine's host name (for example, **fs1.abc.com** or **fs1**)
  – The machine's IP address (for example, **11.22.33.44**)

  Note: If you specify the host name or IP address of the machine, the command executes using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option); unless DFS authorization checking is disabled on the specified machine, a privileged **bos** command issued in this manner fails. If you specify the machine's host name or IP address, the command displays the following message (using the **-noauth** option suppresses the message):

  ```
  bos:  WARNING: short form for server used; no authentication information will be sent
  to the bosserver
  ```

- The **-process** *server_process* option is the process to be created, started, or stopped. The following names are recommended for DFS server processes, but a process can be given any name:
  – **ftserver**: The Fileset Server process
  – **flserver**: The Fileset Location Server process
  – **upclient**: The client portion of the Update Server that transfers binary files (such as those for server processes) from *dcelocal***/bin** and transfers configuration files (such as administrative lists) from *dcelocal***/var/dfs** on the System Control machine
  – **upserver**: The server portion of the Update Server
  – **repserver**: The Replication Server process
  – **bakserver**: The Backup Server process
- The **-cmd** *cmd_line* option specifies the commands and parameters that the BOS Server uses to create a process. For a **simple** process, only one command line specifying the binary file's complete pathname is necessary. This can be the pathname of a DFS command or any other command to be executed. For

example, the command ″ *dcelocal*/**bin/fts clonesys**″ backs up every fileset in the file system. As this example shows, you must enclose the parameter in ″ ″ (double quotes) if it contains spaces, and you must specify the complete pathname for the command.

For a **cron** process, *cmd_line* specifies the following two command parameters:

– The *first parameter* is the command that the BOS Server executes. As with the sole parameter for a **simple** process, this parameter can be the complete pathname of the binary file for a DFS command or any other command to be executed. As the example for the **simple** process shows, you must enclose the parameter in double quotes if it contains spaces, and you must specify the complete pathname for the command.

– The *second parameter* specifies the time at which the BOS Server is to execute the command. This parameter must also be surrounded with double quotes if it contains spaces. Valid values are

- **never**

  The command does not execute, but the process entry remains in the **BosConfig** file.

- **now**

  The command executes immediately, but it never executes again; the process entry is removed from the **BosConfig** file after the command is executed.

- A specific day of the week at a specific time (″ *day hh:mm*″). The command executes weekly at the specified day and time.

- A specific time (*hh:mm*). The command executes daily at the specified time.

  If you specify a day, it must appear first, in lowercase letters. You can enter either the entire name or just the first three letters; for example, **sunday** or **sun**. When indicating a time, separate hours from minutes with a colon. You can use 24-hour time or 1:00 through 12:00 with **am** or **pm** (for example, **14:30** or ″**2:30 pm**″). You must enclose the entry in ″ ″ (double quotes) if it contains spaces (for example, ″**sun 2:30 pm**″).

• The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.

• The **-localauth** option directs the **bos** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, */.../abc.com/hosts/fs1/dfs-server*. Do not confuse a machine’s DFS server principal with its unique **self** identity. (See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for information about DFS server principals.)

  Use this option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

The **-noauth** and **-localauth** options are always optional.

Standard Commands and Operations

Some of the following commands and operations are described in many places in this chapter; others can prove useful when the operations in this chapter are performed. If a command or operation is described in detail here, it is not described in depth in later sections of this chapter where it is used.

Determining Administrative Privilege

To perform the majority of **bos commands**, the issuer must be listed in the **admin.bos** file on the machine used in the command. To determine the members of a list, issue the **bos lsadmin** command:

$ bos lsadmin -server *machine* **-adminlist** *admin.bos*

The **-adminlist admin.bos** option specifies that members of the **admin.bos** file are to be listed.

Examining Log Files

The **bosserver** process and most of the server processes it monitors generate log files. The log files record execution messages and error messages generated by the server processes as they execute. By default, the processes write the files to the *dcelocal***/var/dfs/adm** directory, although some server processes can be instructed to write their log files to a different directory. A list of the log files and the processes that write them follows:

- The **BakLog** file is generated by the Backup Server process on each Backup Database machine.
- The **BosLog** file is generated by the BOS Server process on each server machine.
- The **DfsgwLog** file is generated by the Gateway Server process on each Gateway Server machine.
- The **FlLog** file is generated by the Fileset Location Server process on each Fileset Database machine.
- The **FtLog** file is generated by the Fileset Server process on each File Server machine.
- The **RepLog** file is generated by the Replication Server process on each server machine.
- The **UpLog** file is generated by the **upserver** process on each server machine that is running the server portion of the Update Server.

The **bos getlog** command can be used to examine any of these log files, including the **.old** versions created by the associated server processes. By default, the command looks in the *dcelocal***/var/dfs/adm** directory for the log file that it is to display. It is not necessary to specify the full pathname of a log file if it resides in the default directory. However, if the file resides elsewhere, the full pathname of the log file must be provided.

In addition, no privilege is necessary to view a log file that resides in the default directory. If the file resides in a different directory, the issuer of the command must be listed in the **admin.bos** file on the machine on which the file is located, which is specified by the -server option.

$ bos getlog-server *machine* **-file** *log_file*

The **-file** *log_file* option specifies the log file that is to be displayed. A simple filename is sufficient for a log file that resides in the */.../dcelocal***/var/dfs/adm** directory. A full pathname is required for a log file that resides in a different directory.

Creating and Starting Processes

To start a new process on a server machine, use the **bos create** command to alter the **BosConfig** file. This adds a process entry for the new process to the **BosConfig** file and sets the status flag for the process to **Run** in both the file and the BOS Server's memory, making the effect immediate. You can use the command to create both **simple** and **cron** processes.

The server process name included in this command is used by the BOS Server to reference the process. It is also used in any subsequent **bos** commands that require a process name. The BOS Server adds it to the **BosConfig** file when it creates the process's entry. The name does not appear in process listings generated with the **ps** command or its equivalent.

Creating and Starting a simple Process

To create and start a **simple** process, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the process is to be started. If necessary, issue the **bos lsadmin** command to check.
2. Create an entry for the **simple** process in the **BosConfig** file, and start it:

   $ bos create -server *machine* **-process** *server_process* **-type** \
   simple **-cmd** *cmd_line*...

   The **-type simple** option specifies this as a **simple** process.

   Following is an example **simple** process entry named **flserver** on the machine named **fs1**:

   $ bos create /.../abc.com/hosts/fs1 flserver simple */.../dcelocal***/bin/flserver**

Creating and Starting a cron Process

To create and start a **cron** process, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the process is to be started. If necessary, issue the **bos lsadmin** command to check.
2. Create an entry for the **cron** process in the **BosConfig** file, and start it:

   $ bos create -server *machine* **-process** *server_process* **-type** \
   cron **-cmd** *cmd_line*...

   The **-type cron** option specifies this as a **cron** process.

   Following is a sample **cron** process entry named **backup** on the machine named **fs1**. The **-localauth** option allows the unauthenticated process to use the DFS server principal of **fs1** to execute the privileged **fts clonesys** command.

```
$ bos create /.../abc.com/hosts/fs1 backup cron  \
"dcelocal/bin/fts clonesys -s /.../abc.com/hosts/fs1-localauth " 5:30
```

Listing Status and Machine Information

Use the **bos status** command to check the processes that are running on a server machine. The command causes the BOS Server to probe and determine the status of each process on the machine. It then displays output about the status of each process. It also displays appropriate messages if DFS authorization checking is disabled on the machine or if the machine's *dcelocal* directory or its contents are not protected appropriately.

The process information provided by the **bos status** command enables you to determine the role of the server machine (File Server machine, System Control machine, Binary Distribution machine, Fileset Database machine, Backup Database machine, or multiple machine roles). When you are using the **bos status** command to determine server machine roles, include the **-long** option to provide more detailed output.

Checking the Statuses of Processes on a Server Machine

Enter the **bos status** command to check the statuses of the processes on a server machine. Use the **-process** option to display the statuses of specific processes on the specified server machine, or omit the option to display the statuses of all processes on the machine.

$ bos status -server *machine* **[-process** *server_process***...] [-long]**

The **-long** option indicates that more detailed information about the specified processes is to be displayed.

The command first displays the following line if DFS authorization checking is disabled on the machine (it does not display the line if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

It then displays the following line if the BOS Server finds that the *dcelocal* directory or a directory or file beneath it on the machine has protections that the BOS Server believes are inappropriate:

```
Bosserver reports inappropriate access on server directories.
```

The BOS Server displays this message if the UNIX mode bits on the *dcelocal* directory and its contents do not enforce certain protections; for example, the message can indicate that users who should not be able to write to the *dcelocal* directory and its subdirectories have write access. The BOS Server also displays the message if a directory or file is not owned by the appropriate user (for example, **root**).

The BOS Server displays the message as a courtesy to the user. It does nothing to change the protections, nor does it fail if the protections are violated. (See the description of the **bos status** command in the Reference Part of this guide and reference for information about the protections the BOS Server wants to see enforced.)

The command then displays status information about the processes on the machine. The possible statuses for any process include

- `currently running normally`

  For a **simple** process, this means it is currently running; for a **cron** process, this means it is scheduled to run.

- `temporarily enabled`

  The status flag for the process in the **BosConfig** file is **NotRun**, but the process has been enabled with the **bos startup** or **bos restart** command.

- `temporarily disabled`

  Either the **bos shutdown** command was used to stop the process, or the BOS Server quit trying to restart the process, in which case the message `stopped for too many errors` also appears.

- `disabled`

  The status flag for the process in the **BosConfig** file is **NotRun**, and the process has not been enabled.

- `has core file`

  The process failed or produced a core file at some time. This message can appear with any of the other messages. Core files are stored in *dcelocal*/**var/dfs/adm**. The name of the core file indicates the process that failed; for example, **core.ftserver**.

The output for a **cron** process includes an auxiliary status message, reporting when the command is next scheduled to execute.

The following additional information is displayed when the **-long** option is used:

- The process type (**simple** or **cron**).
- How many **proc start** procedures occurred; **proc start** procedures occur when the process is started or restarted by the current BOS Server.
- The time of the last **proc start**.
- The exit time and error exit time when the process last failed. This appears only if the process failed while the BOS Server was running. (Provided the BOS Server was running both when the process was started and when it failed, the BOS Server can provide this information for any process that has an entry in the **BosConfig** file.)
- The command and its options that are used by the BOS Server to start the process.

The following examples show two executions of the **bos status** command on the same server machine. The first example shows the output displayed when the **-long** option is omitted from the command.

$ bos status /.../abc.com/hosts/fs4

```
Instance ftserver, currently running normally.
```

The second example shows the output displayed when the **-long** option is included with the command.

$ bos status /.../abc.com/hosts/fs4 -long

```
Instance ftserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:02 1991 (1 proc starts)
Parameter 1 is '/dcelocal/bin/ftserver'
```

Determining Server Machine Roles

The following instructions can help you use the **bos status** command to determine which server machines are filling the various machine roles in your cell or domain. The instructions assume that your cell is configured according to the installation and configuration instructions for your system; for example, they assume that all machines except the System Control machine are running a client portion of the Update Server that references the */dcelocal***/var/dfs** directory on the System Control machine. If your server machines are not configured in this manner, these instructions may not help you determine the roles of the machines.

To determine whether a server machine is a System Control machine, a Binary Distribution machine, or neither of the two types of machines, issue the **bos status** command on the machine with **upserver** as the argument for the **-process** option. The output from the command indicates only whether the machine is a System Control machine, a Binary Distribution machine, or neither of the two; a machine that fits neither of the two roles can be a File Server machine, a Fileset Database machine, a Backup Database machine, or any combination of the three.

To learn which machine is the System Control machine, issue the **bos status** command on any server machine, using **upclient** as the argument for the **-process** option. The output for the **upclient** process used to obtain administrative lists from the System Control machine includes the **upclient** command used to start the process. The first parameter of the command is the name of the System Control machine; the second parameter is the pathname to the administrative lists on that machine; for example, */dcelocal***/var/dfs**.

To learn which machine is a Binary Distribution machine, issue the **bos status** command on a server machine of the CPU/OS type you wish to check, again using **upclient** as the argument for **-process** . The output for the **upclient** process used to obtain binary files from the Binary Distribution machine includes the **upclient** command used to start the process. The first parameter of the command is the name of the Binary Distribution machine; the second parameter is the pathname to the binary files on that machine; for example, */dcelocal***/bin**.

When using the **bos status** command to determine machine roles, always use the **-long** option to display more detailed information about the specified processes. You must use the **-long** option to determine the exact role of a server machine.

The following examples illustrate how to determine whether a machine is a System Control machine or a Binary Distribution machine. The output for a server machine that is neither a System Control machine nor a Binary Distribution machine displays that no **upserver** is running.

$ bos status /.../abc.com/fs1 upserver -long

```
bos: failed to get instance info for 'upserver' (no such entity)
```

The output for a System Control machine includes references to the **upserver** process and the */dcelocal***/var/dfs** directory, where administrative lists are stored.

$ bos status /.../abc.com/fs2 upserver -long

```
Instance upserver, (type is simple) currently running normally.
Process last started at Mon Nov 4 05:23:54 1991 (1 proc starts)
Parameter 1 is '/dcelocal/bin/upserver /dcelocal/var/dfs'
```

The output for a Binary Distribution machine includes references to the **upserver** process and the *dcelocal***/bin** directory, where binary files for processes and programs are stored.

$ bos status /.../abc.com/fs3 upserver -long

```
Instance upserver, (type is simple) currently running normally.
Process last started at Mon Nov 4 05:16:31 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/upserver dcelocal/bin'
```

Stopping and Removing Processes

You can stop a process by using the **bos stop** command to set its status flag to **NotRun** in both the BOS Server's memory and the **BosConfig** file. The process then appears as

```
disabled
```

in the output from the **bos status** command. The entry remains in the file, but it does not run again until you issue the **bos start** command, which changes its status flag to **Run** in *both* the memory and the **BosConfig** file. You can also issue the **bos startup** command to run the process by changing its status flag *only* in memory.

To halt a process temporarily (for example, to perform maintenance or make alterations to a configuration), use the **bos shutdown** command to change the process's status in the BOS Server's memory to **NotRun**. The effect is immediate and remains until you again change the memory state or until the BOS Server restarts, at which time it consults the **BosConfig** file and sets the memory state to match the information in the file.

After you stop a process with the **bos stop** command, you can remove it from the **BosConfig** file with the **bos delete** command. It then no longer appears in the output from the **bos status** command. You must use the **bos stop** command to stop a process (**simple** or **cron**) whose status is **Run** before you use the **bos delete** command to remove it from the **BosConfig** file. An error occurs if the status of a process being deleted is **Run** when the **bos delete** command is issued.

CAUTION:
Do not temporarily stop a database server process on all machines simultaneously. This would make the database totally unavailable.

Stopping Processes by Changing Their Status Flags to NotRun

To stop processes by changing their status flags to **NotRun**, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be stopped. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos stop** command to stop the processes by changing their status flags in the **BosConfig** file and in the BOS Server's memory to **NotRun**:

$ bos stop -server *machine* **-process** *server_process***... [-wait]**

The **-wait** option causes the command shell prompt to remain absent until the processes have stopped. If omitted, the prompt immediately returns, even if the processes have not yet stopped.

Stopping Processes Temporarily

To stop processes temporarily, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be stopped. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos shutdown** command to stop the processes by changing their status flags in the BOS Server's memory to **NotRun**:

   ```
   $ bos shutdown -server machine [-process server_process...] [-wait ]
   ```

   The **-process** *server_process* option specifies each server process to be stopped. Omit this option to stop all processes except the BOS Server.

   The **-wait** option causes the command shell prompt to remain absent until the processes have stopped. If omitted, the prompt immediately returns, even if the processes have not yet stopped.

Removing Processes from the BosConfig File

To remove processes from the **BosConfig** file, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine from which the process is to be removed. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos stop** command to stop the processes by changing their status flags in the **BosConfig** file and in the BOS Server's memory to **NotRun**; you must also do this for **cron** processes, even though they do not run continuously.

   ```
   $ bos stop -server machine -process server_process... [-wait ]
   ```

   The **-wait** option causes the command shell prompt to remain absent until the processes have stopped. If omitted, the prompt immediately returns, even if the processes have not yet stopped.
3. Remove the processes from the **BosConfig** file with the **bos delete** command:

   ```
   $ bos delete -server machine -process server_process...
   ```

Starting Processes

When starting processes, you can use the **bos start** command to change their status flags to **Run** in both the **BosConfig** file and in the BOS Server's memory. You can also start processes that are temporarily disabled (processes that have a status of **Run** in the **BosConfig** file but a status of **NotRun** in memory) by using the **bos startup** command and changing only the memory state to **Run**. You can use the **bos startup** command to change a process's status in memory to **Run** even if its status in the **BosConfig** file is **NotRun**; thus, you can use the **bos startup** command to run tests on a server process without enabling it permanently.

A newly started process is a completely new instance; if you install new binaries during the time a process is shut down, they are used when you issue **bos start** or **bos startup**. If an instance of a process is already running, the only effect of these commands is to ensure that the process's status flag is set to **Run** in memory and, if **bos start** is used, in the **BosConfig** file; you must issue the **bos restart** command to start a new instance of the process.

Starting Processes by Changing Their Status Flags to Run

To start processes by changing their status flags to **Run**, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos start** command to start the processes by changing their status flags in the **BosConfig** file and in memory to **Run**:

   bos start -server *machine* **-process** *server_process*...

Starting All Stopped Processes That Have BosConfig Flags of Run

To start all stopped processes that have status flags of **Run** in the **BosConfig** file, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Use the **bos startup** command to start each process that has a status flag of **Run** in the **BosConfig** file; this changes each process's status flag in the BOS Server's memory from **NotRun** to **Run**. Each process's status flag in the **BosConfig** file remains the same.

   $ bos startup -server *machine*

Starting Specific Temporarily Stopped Processes

To start processes that were temporarily stopped, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Use the **bos startup** command to start each specified process by changing its status flag in the BOS Server's memory to **Run**. Each process's status flag in the **BosConfig** file remains unchanged.

   $ bos startup -server *machine* **-process** *server_process*...

Restarting Processes

You may sometimes need to stop and then restart a process (for example, to load a new binary file immediately rather than wait for the BOS Server to perform its daily check for new files, which is described in "Setting Scheduled Restart Times" on page 126). You can use the **bos restart** command to stop and restart any or all processes on a server machine, including the BOS Server itself. The **bos restart** command can be used to restart only those processes already controlled by the BOS Server. It does not change the status flag for a process in the **BosConfig** file.

CAUTION:
Restarting some processes can cause a service outage. You should schedule these outages for times of low usage on the system.

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the processes are to be restarted. If necessary, issue the **bos lsadmin** command to check.

2. There are three ways to use the **bos restart** command: you can stop and restart specific processes; you can stop and restart all processes, *including* the BOS Server; or you can stop and restart all processes *except* the BOS Server.

   To stop and restart specific processes, use the **-process** option with the **bos restart** command. Specify the name of each server process to be stopped and restarted. The BOS Server stops and immediately restarts all specified processes, regardless of their status flags in the **BosConfig** file.

   bos restart -server *machine* **-process** *server_process*...

   To restart all processes including the BOS Server, use the **-bosserver** option with **bos restart** the command. The BOS Server stops all processes, including itself. A new BOS Server immediately starts; it then restarts all processes with the status flag **Run** in the **BosConfig** file.

   bos restart -server *machine* **-bosserver**

   The **-bosserver** option indicates that the BOS Server on **-server** is to stop all processes, including itself; a new BOS Server starts, restarting all processes with the status flag **Run**.

   To stop and restart all processes except the BOS Server, omit both the **-process** and **-bosserver** options from the **bos restart** command. The BOS Server stops all processes except itself. It then immediately restarts all processes with the status flag **Run** in the **BosConfig** file.

   $ bos restart -server *machine*

Installing Process Binary Files

Binary files for DFS server processes are stored on the local disk of each server machine. By default, the files are stored in the *dcelocal***/bin** directory. The Binary Distribution machine for each CPU/OS type in a cell houses the master versions of the binary files for its machine and operating system type in this same local directory. The files can be stored in a different directory on any machine, but it avoids potential confusion if they are stored in the default directory on all machines.

The **bos install** command can be used to install a new process binary file on a server machine. It should be used to install new binary files only on Binary Distribution machines. The files are then distributed from each Binary Distribution machine to other server machines of the same CPU/OS type via the Update Server. By default, the **upclient** process on each server machine checks the Binary Distribution machine of its CPU/OS type for new (or different) versions of binary files every 5 minutes; if it finds that versions of files installed on the Binary Distribution machine are different from those on the local machine, it automatically copies the files to its local machine via the **upserver** process on the Binary Distribution machine.

Do not install new binary files on a server machine other than the Binary Distribution machine. The binary files are overwritten the next time the **upclient** process on the machine copies versions of files from the Binary Distribution machine of its CPU/OS type.

The **bos install** command preserves former versions of files in the installation directory by assigning **.BAK** and **.OLD** extensions as follows:

- If a current version of the file exists, the command adds a **.BAK** extension to its name.

- If a **.BAK** version of the file exists, the command changes its extension to **.OLD** before giving the current version a **.BAK** extension.
- If **.BAK** and **.OLD** versions of the file exist and the current **.BAK** version is less than 7 days old, the current version of the file overwrites the current **.BAK** version, but the **.OLD** version remains unchanged.
- If **.BAK** and **.OLD** versions of the file exist and the current **.BAK** version is at least 7 days old, the current **.BAK** version overwrites the current **.OLD** version, and the current version of the file overwrites the current **.BAK** version. Use the **bos getdates** command to examine the time stamps of current, **.BAK**, and **.OLD** versions of binary files to determine when they were installed.

The **bos install** command installs all files with the UNIX mode bits set to **755**, regardless of the mode bits associated with a version of the file that currently exists in the installation directory. These permissions are subject to the **umask** associated with the BOS Server on the machine on which the files are installed. The mode bits associated with the current version of the file are preserved when it becomes the **.BAK** version. The **bos install** command neither preserves nor manipulates the Access Control List (ACL) permissions of a file installed from or to a DCE LFS fileset.

The **bos uninstall** command replaces the current version of a binary file with the next-oldest version of the file: the **.BAK** version, if it exists; otherwise, the **.OLD** version. If both the **.BAK** and **.OLD** versions exist, the **.OLD** version replaces the **.BAK** version when the latter becomes the current version. The **bos uninstall** command's **-all** option can be used to remove all versions of a binary file.

The **bos prune** command can be used to remove only the **.BAK** and **.OLD** versions of binary files from the *dcelocal***/bin** directory. The **bos prune** command can also be used to remove core files, which are generated when processes monitored by the BOS Server go down, from the *dcelocal***/var/dfs/adm** directory.

After new versions of binary files for processes controlled by the BOS Server are installed on a machine, you can use the **bos restart** command to begin using them immediately. Otherwise, the new versions are not used until the next new binary restart time (specified in the *dcelocal***/var/dfs/BosConfig** file) of the BOS Server on the machine. (See "Setting Scheduled Restart Times" on page 126 for detailed information about checking and setting scheduled restart times.)

Installing New Binary Files

To install new binary files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary files are to be installed. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos install** command to install a new version of each specified binary file:

   ```
   $ bos install -server machine -file binary_file... \
    [-dir  alternate_dest]
   ```

   The **-file** *binary_file* option specifies the pathname of each binary file to be installed on the machine specified with **-server**. For each file, specify either a full or a relative pathname; relative pathnames are interpreted relative to the current working directory. An installed file replaces a file of the same name.

The **-dir** *alternate_dest* option specifies the pathname of the directory on **-server** in which all specified files are to be installed. Omit the **-dir** option to install the files in the default directory, *dcelocal***/bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

Replacing Binary Files with Older Versions

To replace binary files with older versions of the files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary files reside. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos uninstall** command to replace the current version of each specified binary file with its next-oldest (**.BAK** or **.OLD**) version:

```
$ bos uninstall -server machine -file  binary_file... \
 [-dir  alternate_dest] [-all ]
```

The **-file** *binary_file* option specifies the name of each binary file to be replaced with its next-oldest version. Each specified file is replaced with its **.BAK** version, if it exists; otherwise, it is replaced with its **.OLD** version. If both the **.BAK** and **.OLD** versions exist, the **.OLD** version also replaces the **.BAK** version. All specified files must reside in the same directory (*dcelocal***/bin** or an alternate directory specified with the **-dir** option). Specify only filenames; the command ignores all but the final component of a pathname.

The **-dir** *alternate_dest* option provides the pathname of the directory in which all specified files reside. Omit the **-dir** option if the files reside in the default directory, *dcelocal***/bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

The **-all** option indicates that all versions (current, **.BAK**, and **.OLD**) of each specified file are to be removed.

Checking the Time Stamps on Binary Files

Enter the **bos getdates** command to determine when binary files were installed:

```
$ bos getdates -server machine -file binary_file... \
 [-dir alternate_dest]
```

The **-file** *binary_file* option specifies the name of the current version of each binary file whose time stamps are to be displayed. The time stamps on the current, **.BAK**, and **.OLD** versions of each file are displayed. All specified files must reside in the same directory, *dcelocal***/bin**, or an alternate directory specified with the **-dir** option. Specify only filenames; the command ignores all but the final component of a pathname.

The **-dir** *alternate_dest* option specifies the pathname of the directory in which all specified files reside. Omit this option if the files reside in the default directory, *dcelocal***/bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

Removing Old Binary and Core Files

To remove old binary and core files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary and core files reside. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos prune** command to remove old versions of **.BAK** files, **.OLD** files, core files, or any combination of the files from the *dcelocal***/bin** and *dcelocal***/var/dfs/adm** directories:

   ```
   $ bos prune -server machine [-bak ] [-old ] [-core ] [-all ]
   ```

   The **-bak** option specifies that all **.BAK** files are to be removed from *dcelocal***/bin**. Use this option and optionally **-old**, **-core**, or both, or use **-all**.

   The **-old** option specifies that all **.OLD** files are to be removed from *dcelocal***/bin**. Use this option and optionally **-bak**, **-core**, or both, or use **-all**.

   The **-core** option specifies that all core files are to be removed from *dcelocal***/var/dfs/adm**. Use this option and optionally -bak, **-old**, or both, or use **-all**.

   The **-all** option specifies that all **.BAK** and **.OLD** files are to be removed from *dcelocal***/bin** and all core files are to be removed from *dcelocal***/var/dfs/adm**. Use this option or use some combination of **-bak**, **-old**, and **-core**.

Removing All Versions of Binary Files

To remove all versions of binary files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary files reside. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos uninstall** command and include its **-all** option to remove all versions of each specified binary file:

   ```
   bos uninstall -server machine -file binary_file... \
    [-dir  alternate_dest] [-all ]
   ```

   The **-file** *binary_file* option specifies the name of each binary file to be removed. The current, **.BAK**, and **.OLD** versions of each file are removed. All specified files must reside in the same directory, *dcelocal***/bin**, or an alternate directory specified with the **-dir** option. Specify only filenames; the command ignores all but the final component of a pathname.

   The **-dir** *alternate_dest* option specifies the pathname of the directory in which all specified files reside. Omit this option if the files reside in the default directory, *dcelocal***/bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

   The **-all** option indicates that all versions (current, **.BAK**, and **.OLD**) of each specified file are to be removed.

Setting Scheduled Restart Times

The BOS Server performs two types of scheduled restarts: a general restart and a new binary restart. During a general restart, the BOS Server first restarts itself (using a new binary file, if one exists); it then restarts all other server processes on its machine that have a status flag of **Run** in the **BosConfig** file. It is recommended

that the general restart time be set as a weekly time; by default, the BOS Server performs this type of restart weekly, on Sunday at 4:00 a.m.

During a new binary restart, the BOS Server checks for newly installed binary files. Binary files are installed on Binary Distribution machines with the **bos install** command, after which they are propagated to other machines by the Update Server. If a new version of a process's binary file was installed in *dcelocal***/bin** after the process last started on the server machine, the BOS Server restarts the process so that the new instance of the binary file is used. It is recommended that the new binary restart time be specified as a daily time; by default, the BOS Server executes this type of restart daily, at 5:00 a.m.

The default general and new binary restart times are set for early morning, when system usage is typically lowest. The **BosConfig** file on every server machine records the two restart times. This is a local file, so the information can be different for different machines. You can check and reset both time settings with the **bos getrestart** and **bos setrestart** commands.

You can set a restart time as a day and time or as just a time. When including a day, specify the day first, in lowercase letters; you can enter the entire name or just the first three letters (for example, **sunday** or **sun**). When indicating a time, separate hours from minutes with a **:** (colon); you can use 24-hour time or 1:00 through 12:00 with **am** or **pm** (for example, **14:30** or ″**2:30 pm**″). You must enclose the entire entry in ″ ″ (double quotes) if it contains spaces (for example, ″**2:30 pm**″ or ″**sun 14:30**″).

You can also set a restart time as **never** or **now**. The setting **never** indicates that the BOS Server does not perform the indicated type of restart. For a restart time, the setting **now** is equivalent to specifying the current day and time.

CAUTION:
Never edit the restart times in the BosConfig file directly; use the bos setrestart command only. If you edit the restart times directly, the BOS Server does not recognize the new times until it is restarted and again reads the BosConfig file.

Checking the Current Restart Times

To check the current time settings, issue the **bos getrestart** command:

```
$ bos getrestart -server machine
```

Following is an example of the output from this command:

```
$ bos getrestart /.../abc.com/hosts/fs3
Server /.../abc.com/hosts/fs3 restarts at sun 4:00 am
Server /.../abc.com/hosts/fs3 restarts for new binaries at 5:00 am
```

Setting the General Restart Time

To set the general restart time, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine whose general restart time is to be set. If necessary, issue the **bos lsadmin** command to check.
2. Set the general restart time by issuing the **bos setrestart** command with the **-general** option:

```
$ bos setrestart -server machine -general time
```

The **-general** option identifies this as the general restart time, not the new binary restart time; *time* is the time at which the BOS Server is to restart itself and the other processes it controls.

Setting the New Binary Restart Time

To set the new binary restart time, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine whose new binary restart time is to be set. If necessary, issue the **bos lsadmin** command to check.
2. Set the new binary restart time by issuing the **bos setrestart** command with the **-newbinary** option:

   ```
   $ bos setrestart -server machine -newbinary time
   ```

   The **-newbinary** option identifies this as the new binary restart time, not the general restart time; *time* is the time at which the BOS Server is to check for new binary files in */dcelocal/***bin**.

Rebooting a Server Machine

Note: Consult other DCE component documentation to determine the impact of rebooting on other DCE components.

Rebooting a server machine, while not difficult, should never be the first method used to solve DFS-related problems. You should reboot a server machine only if no other recourse is available, such as when a process that is not controlled by the BOS Server fails. Rebooting causes a service outage. If the machine being rebooted is the only Fileset Database machine, it can make the entire file system unavailable to all users; if the machine is a File Server machine, people using filesets located only on that machine (for example, user filesets) cannot access those filesets.

To prepare a server machine for powering down, you can issue the **bos shutdown** command to have the BOS Server shut down the other server processes that are running on the machine; the BOS Server does not shut itself down—it terminates correctly when you turn off the machine. You can then reboot the machine by issuing the machine's **reboot** command (or its equivalent).

You can reboot a machine from either the local console or the console of a remote machine (via **telnet** or an appropriate program). The two approaches are essentially the same, with the exception that rebooting from the local console lets you track the status of the reboot as it occurs, which you cannot do with remote rebooting. Regardless of the reboot method you use, server processes restart automatically after the reboot if the machine's initialization file (**/etc/rc** or its equivalent) contains the following instruction to restart the BOS Server:

```
dcelocal/bin/bosserver
```

Note:

1. On AIX, the steps that follow to reboot a server machine are unnecessary. The **shutdown** command shuts down DCE and DFS services by running the **stop.dfs** and **stop.dce** commands.
2. **/etc/inittab** should have a line to restart DCE and DFS. For example,

```
/etc/dce/rc.dce all
```

will call **start.dce** and **start.dfs**.

3. For information about configuring AIX so it automatically restarts DCE and DFS services at system restart time, see *IBM DCE for AIX, Version 2.2: Quick Beginnings*.
4. Consult other DCE component documentation to determine the impact of rebooting on other DCE components.

To reboot a server machine, do the following:

1. *To reboot from the console of a remote machine*, open a remote connection to the machine you want to reboot (using **telnet** or an appropriate program). To reboot from the local console of the machine you want to reboot, omit this step.
2. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine to be rebooted. If necessary, issue the **bos lsadmin** command to check.
3. Issue the **bos shutdown** command to prepare to power down the machine to be rebooted. This command directs the BOS Server to shut down the other DFS server processes that are running on the machine by changing their status flags in the BOS Server's memory to **NotRun**. The BOS Server does not shut itself down; it terminates safely when you turn off the machine. Include the **-wait** option to be sure that all processes have stopped before performing the next step.

   $ bos shutdown -server *machine* **-wait**

   The **-wait** option causes the command shell prompt to remain absent until the processes are stopped. If the **-wait** option is omitted, the prompt returns immediately, even if the processes are not yet stopped.
4. Log in as **root** in the native UNIX file system of the machine to be rebooted. For example:

   $ su root

   Password: *root_password*
5. Issue the appropriate reboot command (**/usr/sbin/shutdown** or its equivalent) for the machine to be rebooted. For example:

```
/usr/sbin/shutdown -Fr
```

Chapter 6. Making Filesets and Aggregates Available

In the DCE Local File System (DCE LFS), a fileset is defined as a collection of related files that are organized into a single, easily managed unit. Because DCE LFS filesets are usually smaller in size than standard file systems, and because each DCE LFS aggregate can house multiple DCE LFS filesets, DCE LFS filesets are easily moved between File Server machines to facilitate load balancing across the network. It is also easy to place read-only copies (replicas) of DCE LFS filesets on different machines in your cell. These multiple copies prevent machines from becoming overburdened with requests for files from popular filesets.

In other operating systems, a file system typically occupies more disk space and is tied to a physical location. In addition, non-DCE LFS file systems (non-DCE LFS filesets) cannot be replicated in DFS.

This chapter provides detailed information about how to create, replicate, and back up DCE LFS filesets, and how to mount DCE LFS and non-DCE LFS filesets for use in the DCE namespace. It also explains how to export aggregates and partitions. (See “Chapter 7. Managing Filesets” on page 179 for information on the tasks involved in the use and maintenance of filesets.)

Note:

1. This guide uses the term *non-LFS* to refer to *non-DCE LFS* filesets and aggregates. For AIX, this can be AIX Journaled File Systems or AIX CD-ROM File Systems.
2. DCE 2.2 for AIX provides additional commands to perform initial DFS configuration tasks and some DFS administrative tasks. For more information, see *IBM DCE for AIX, Version 2.2: Quick Beginnings*. For specific information about the **config.dfs**, **mkbutc.dfs**,**mkfilesys.dfs**, **rmbutc.dfs**, **rmfilesys.dfs** and **unconfig.dfs**commands, see the Reference part of this guide and reference. For more information about using the AIX SMIT utility to configure and manage DFS, see *IBM DCE for AIX, Version 2.2: Quick Beginnings*.
3. Many tasks described in this section are performed for you when you configure DFS servers using the **config.dfs**, **mkfilesys.dfs**or the AIX SMIT utility. The steps are provided here for illustration purposes. You do not need to perform them.

An Overview of Filesets

A DCE LFS fileset is a hierarchical grouping of files that is managed as a single unit. A DCE LFS aggregate is a disk partition that is modified to include the DCE LFS metadata structure that supports DCE Access Control Lists (ACLs), multiple DCE LFS filesets, logging, and other fileset-related operations.

Using DFS, you can share information stored on the local disks of different machines by exporting aggregates and partitions from the machines. Exporting an aggregate or partition makes the filesets contained on it available in the DCE namespace. With the DCE LFS, you can export multiple filesets from one aggregate. Because non-LFS partitions do not support the enhancements that are supported on DCE LFS aggregates, and because you can store only one fileset on a non-LFS partition, you can export only one non-LFS fileset per nonLFS partition.

Figure 6 illustrates the structural differences between the DCE LFS and other file systems. The partitioning structure in the DCE LFS features aggregates, each of which can store multiple DCE LFS filesets; the partitioning structure in other file systems has partitions that can house only a single non-LFS file system each. (Note that the disks in both structures include a non-LFS boot partition.)

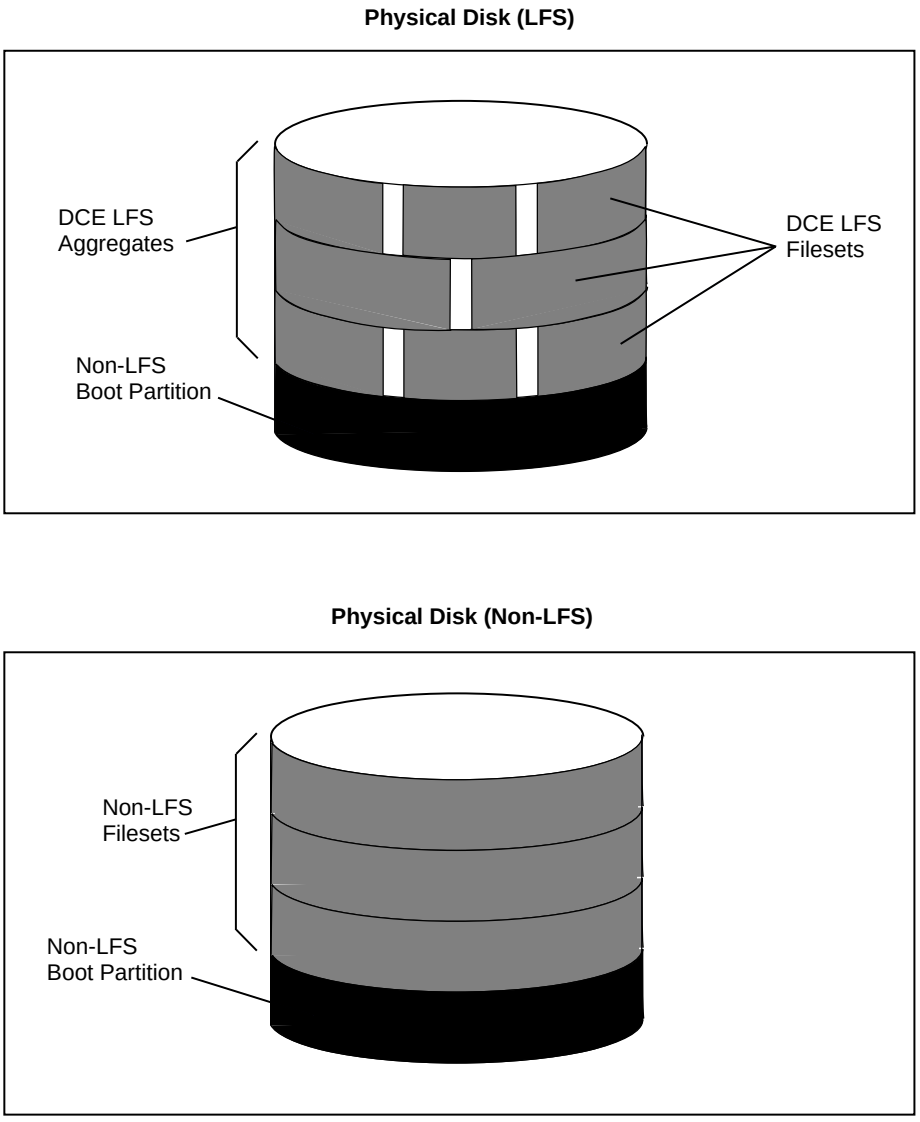


Figure 6. Comparison of DCE LFS and non-LFS Disk Partitioning Structures

Creating and Using Filesets

Before you can create a DCE LFS fileset, you must first use the **newaggr** command to initialize, or format, the disk partition on which the aggregate is to reside; the **newaggr** command is comparable to the UNIX **newfs** command. You must then export the aggregate with the **dfsexport** command to make it available in the DCE namespace.

After the DCE LFS aggregate is initialized and exported, a DCE LFS fileset can be created on it with the **fts create** command. This command also registers the fileset in the Fileset Location Database (FLDB) and obtains a unique ID number for the fileset. To make the contents of a DCE LFS fileset visible in the DCE namespace, enter the **fts crmount** command to create a mount point for the fileset. After the **fts crmount** command is issued, the fileset is automatically attached to the DFS file system and is accessible to authorized DCE users.

On the other hand, when creating a non-LFS fileset, you do not use **fts** commands. Because a disk partition is equal to one non-LFS fileset, you use the **newfs** command (or the command appropriate to your operating system) to initialize the partition on which the non-LFS fileset is to reside. You then use your operating system's **mount** command (or its appropriate equivalent) to mount the partition locally, after which data can be placed on the partition and used locally. Note that, if your vendor has properly modified your local operating system's **mount** command, you can also mount and use a DCE LFS fileset locally.

To make a non-LFS fileset visible in the DCE namespace, you first use the **fts crfldbentry** command to register the fileset in the FLDB and generate a unique ID number for it. You then export the partition on which the fileset resides with the **dfsexport** command and mount the fileset with the **fts crmount** command. The terms *aggregate* and *non-LFS aggregate* can also be used to refer to an exported partition.

The Different Types of DCE LFS Filesets

DCE LFS provides three types of filesets: read/write, read-only, and backup. Non-LFS file systems do not have these different types of filesets. When used with DFS, non-LFS filesets are essentially treated as read/write filesets. However, a partition that houses a non-LFS fileset can be marked as read-only in the local operating system; DFS treats it as a read-only fileset (it cannot be modified), but the fileset does not receive a **.readonly** extension.

Every DCE LFS fileset has a single read/write version, which contains the modifiable versions of the files and directories in that fileset. This version is also referred to as the read/write source because the other fileset types are derived from it via replication and backup operations.

A read-only fileset is an exact copy, or replica, of all of the data in a read/write source fileset when the read-only replica is created. Each read-only fileset is given the same name as its read/write source with an additional **.readonly** extension. Read-only filesets can be placed at various sites in the file system; a site is a specific aggregate on a File Server machine. A read-only fileset cannot be modified by commands such as **mkdir** or **rm** (or their equivalent commands). If the read/write source fileset changes, the read-only versions must be updated to match the changed read/write version; otherwise, they remain unchanged. The update process can be performed manually (via Release Replication) or it can be automated (via Scheduled Replication).

A backup fileset is a clone of a read/write source fileset stored at the same site and with the same name as the source, with the addition of a **.backup** extension. A backup fileset is not the same as a backup *of* a fileset (for example, a copy on tape), but making a backup fileset is often one step in the backup process. (See "Chapter 9. Configuring the Backup System" on page 241 for more information on the backup process.)

Figure 7 illustrates the different types of DCE LFS filesets: read/write, read-only, and backup.

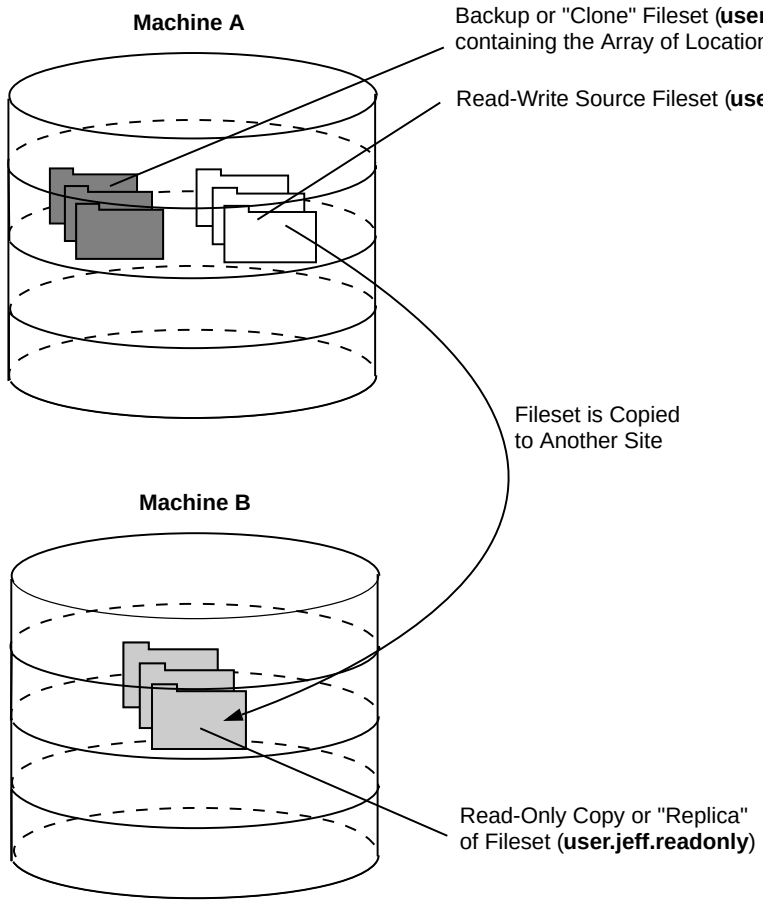


Figure 7. The Different Types of DCE LFS Filesets

Data Sharing Among the Different Types of DCE LFS Filesets

When a backup or read-only fileset occupies the same site (File Server machine and aggregate) as its read/write source fileset, DFS attempts to save disk space by having the filesets share data that is the same across the different types of filesets. This data sharing is accomplished in the following way:

- When the backup or read-only fileset is created, the new fileset is filled with an array of pointers to the data housed by the read/write source.
- The identities of the read/write source and the backup or read-only fileset are then exchanged so that the read/write source becomes the backup or read-only fileset, and the backup or read-only fileset becomes the read/write source.

This technique provides full access to the data via the read/write fileset without requiring that the read/write fileset physically house the data. As long as the read-only or backup fileset remains identical to the read/write source, the disk space occupied by the read/write fileset remains small (because pointers take up much less disk space than the data to which they point). However, as changes are made to the data in the read/write fileset, the amount of space occupied by the read/write fileset increases. This is because the read/write fileset must acquire

additional disk blocks to store changed data; it can no longer simply point to disk blocks housed by the read-only or backup fileset.

Because of this data-sharing arrangement, DFS provides statistics on two types of disk usage for a fileset. The first statistic, quota, identifies the amount of disk space occupied by all of the files and directories in the read/write fileset, *including* those files and directories in the read/write fileset that are actually pointers to disk blocks in the backup or read-only version of the fileset. The second statistic, allocation, identifies the amount of disk space occupied by those files and directories actually housed in the fileset, *excluding* those files and directories that are represented by pointers to disk blocks in another version of the fileset.

Users are concerned with the quota statistics only, because the quota dictates the amount of data that they can store in a read/write fileset. To check a fileset's quota statistics, issue the **fts lsquota** command. (See the Reference part of this guide and reference for information on the **fts lsquota** command.)

Administrators are also concerned with the quota statistics, especially when dealing with users. One of the most common user requests of an administrator is for an increase in the size of the user's fileset quota. To change the size of a fileset's quota, issue the **fts setquota** command. (See "Chapter 7. Managing Filesets" on page 179 for information on changing a fileset's quota.)

Administrators are also concerned with the allocation statistics when they need information on the physical disk usage of a fileset (for example, when moving filesets for load-balancing purposes). To check both the quota and allocation statistics for an individual fileset, issue the **fts lsft** command. To check the quota and allocation statistics for multiple filesets on a File Server machine, issue the **fts lsheader** command. (See "Chapter 7. Managing Filesets" on page 179 for information on the **fts lsft** and **fts lsheader** commands.)

Note: There is no way to set or change the allocation for a fileset. It is automatically set during the creation of the fileset and cannot be changed.

Identifying DCE LFS and Non-LFS Filesets

Every DCE LFS and non-LFS fileset is identified by a unique name and ID number. The following subsections discuss these two forms of fileset identification.

Fileset Names

Every fileset must have a fileset name that is unique within the cell in which it resides. The name is stored in the cell's FLDB. You assign a name to a DCE LFS fileset when you create the read/write version of the fileset and register it in the FLDB with the **fts create** command. You assign a name to a non-LFS fileset when you register the read/write (and only) version of the fileset in the FLDB with the **fts crfldbentry** command.

You can use the following characters in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a through z, and A through Z)
- All numerals (0 through 9)
- The **.** (dot)
- The **–** (dash)

- The _ (underscore)

A fileset name must include at least one alphabetic character or an _ (underscore); it cannot consist of just numbers, dots, and dashes. This allows the system to differentiate the name of the fileset from its ID number.

The name you assign to a fileset can contain no more than 102 characters. This does not include the **.readonly** or **.backup** extension, which is automatically added when a process creates a read-only or backup fileset. Note that the **.readonly** and **.backup** extensions are reserved for read-only and backup filesets, so you cannot specify a fileset name that ends with either of these extensions.

Note: Fileset names can actually be as long as 111 characters—the name of the fileset plus the appropriate **.readonly** or **.backup** extension. However, you can specify only the first 102 characters of the name to accommodate the extensions. This is also true of non-LFS fileset names, even though non-LFS filesets do not need the extensions because they cannot have read-only and backup versions.

With DCE LFS, each user's home directory typically corresponds to a separate fileset. You may find it convenient to name all user filesets **user.** *user_name* (for example, *user.sandy*). It may also be convenient to indicate the type of aggregate in which the fileset is stored (for example, **ufs.fs1** to indicate a non-LFS aggregate from the machine named **fs1**). You may also want to put system binaries into filesets with names that begin with the system type (for example, **rs_aix41.bin**).

When specifying the name of an existing fileset in a DFS command, include the **.readonly** or **.backup** extension, if appropriate, to indicate the read-only or backup version of the fileset. You must include the extension when you wish to perform an operation that affects only that version of a fileset. For example, to delete just the backup version of a fileset, you must add the **.backup** extension to the name of the fileset when you issue the **fts delete** command.

Fileset ID Numbers

Every fileset also has a fileset ID number that, like a fileset name, is unique within the cell in which the fileset resides. When a fileset is registered in the FLDB with the **fts create** or **fts crfldbentry** command, the FL Server allocates it a fileset ID number, which is stored in the FLDB along with its name. Read/write and backup filesets have their own fileset IDs, which are automatically reserved in the FLDB when the read/write source fileset is registered; all read-only copies of the same read/write fileset share a common fileset ID.

Fileset ID numbers are represented as two positive integers separated by a pair of commas. For example, the ID number of the first fileset in the FLDB is **0,,1**. The integer after the commas is then incremented every time a new fileset is created. When the integer after the commas becomes larger than 2^{32}, the integer before the commas becomes 1 and the integer after the commas returns to 0 (zero).

When specifying a fileset ID in a DFS command, you can omit the integer before the commas if it is a 0 (zero); commands that accept a fileset ID number assume that the first integer is 0 (zero) if it is not supplied. In this case, also omit the two commas. For example, the fileset ID number **0,,1** can be entered as **1**.

Tracking Fileset Locations

The Fileset Location Database (FLDB) is maintained by the FL Server. The FLDB records information about the locations of the filesets in a cell. Users do not need to know the location of a fileset to access it; the Cache Manager contacts the FL Server to obtain the location of a requested fileset.

Each read/write fileset has an entry in the FLDB; the entry includes information about the fileset's read-only and backup versions. Read-only and backup filesets normally do not have their own FLDB entries because they share an FLDB entry with the read/write fileset. However, a read-only fileset can have its own entry if its read/write source is removed.

For a DCE LFS fileset, information is also stored in the fileset's header, which is part of the data structure that records the physical addresses on the aggregate of the files in the fileset. It is essential that the FLDB entry and the fileset header be synchronized (that they match). Therefore, all **fts** commands that affect fileset status and the FLDB change both the appropriate FLDB entry and the fileset header. In some rare cases, however, you may need to resynchronize the entries yourself (see "Chapter 7. Managing Filesets" on page 179).

A non-LFS fileset does not have a fileset header, but some information about the fileset is available from the local disk of the machine on which it resides. For example, the fileset ID number of each non-LFS fileset is stored in the *dcelocal*/**var/dfs/dfstab** on the local machine. (See "Exporting a DCE LFS Aggregate" on page 152 for a description of the **dfstab** file.)

A Fileset's FLDB Information

The **fts lsfldb** and **fts lsft** commands display information from a fileset's FLDB entry (the **fts lsft** command also shows information from a fileset's header). Each FLDB entry for a DCE LFS fileset contains the following information:

- The name of the fileset, with a **.readonly** or **.backup** extension, if appropriate.
- The fileset IDs of the read/write, read-only, and backup versions.
- A separate status flag for each of the three versions, indicating whether the version exists at some site. A status of **valid** indicates the version exists at some site; a status of **invalid** indicates the version does not exist at any site. Note that the status of the read-only version is **valid** once a replication site is defined, regardless of whether a replica yet exists at the site.
- The number of sites at which a version of the fileset exists.
- An indicator if the FLDB entry is locked. The indicator is omitted if the entry is not locked.
- The replication parameters that are associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) where read/write (**RW**), read-only (**RO**), or backup (**BK**) versions of the fileset reside.
- For each read-only site, the MaxSiteAge replication parameter defined for that site. (See "Creating Read-Only DCE LFS Filesets" on page 160 for more information about replication parameters.)
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine in the FLDB or **<nil>** if no group owns the server entry.

- For each fileset with advisory RPC authentication bounds, the values for both sets of upper and lower bounds (one set for communications with Cache Managers in the local cell and the other set for communications with Cache Managers in foreign cells).

Because functionality such as replication is not supported for non-LFS filesets, FLDB entries for non-LFS filesets do not contain as much information as entries for DCE LFS filesets. However, information such as the ID number and site of each non-LFS fileset is recorded in the FLDB. The **fts lsfldb** and **fts lsft** commands display this information.

A Fileset's Header Information

A separate fileset header is stored at each site where a version of a DCE LFS fileset exists. The header is part of the data structure that records disk addresses on the aggregate where the files in the fileset are stored. This data structure is a method of grouping all of the files into logical units without requiring that they be stored in contiguous memory blocks. In addition, the header records some of the same information that appears in the FLDB. Therefore, even if the FLDB is unavailable, the **fts** commands can still access the information.

The **fts lsheader** and **fts lsft** commands display information from a fileset's header (the **fts lsft** command also shows information from a fileset's entry in the FLDB). Each fileset header for a DCE LFS fileset contains the following information:

- The name of the fileset, with a **.readonly** or **.backup** extension, if appropriate.
- The fileset ID number.
- The type of fileset (**RW** for read/write, **RO** for read-only, or **BK** for backup).
- Information about the state of the fileset.
- The status flag for the site, including **On-line**, **Off-line**, or an error condition.
- The File Server machine, aggregate name, and aggregate ID number where the fileset resides. This information, while not in the header, is available because it was used to contact the machine that houses the fileset.
- The ID numbers of the parent, clone, and backup filesets that are related to the fileset.
- The ID numbers of the low-level backing and low-level forward filesets that are related to the fileset.
- The version number of the fileset. Every DCE LFS fileset has a distinct version number that increments every time an operation is performed on the fileset or a file it contains. Version numbers have the same format as fileset ID numbers (for example, **0,,25963**).
- The allocation and allocation usage, in kilobytes, of the fileset.
- The quota and quota usage, in kilobytes, of the fileset.
- The day, date, and time that the fileset was created (for read-only and backup versions, this indicates the day, date, and time that the fileset was replicated or backed up).
- The day, date, and time that the contents of the fileset were last updated.

Because non-LFS filesets do not have DCE LFS fileset headers, only such information as the fileset ID number is available from the machine that houses the fileset. The **fts lsheader** and **fts lsft** commands display this information.

Replicating DCE LFS Filesets

Replication is the process of creating one or more read-only copies of the read/write version of a DCE LFS fileset and placing the copies at multiple sites. With replication, you can tailor your system's configuration by making a fileset's contents accessible from more than one File Server machine. As a result, a single machine is not overburdened with requests for popular files, and files are available from more than one machine, which can minimize the effects of a machine failure. Replication is not available for non-LFS filesets.

You can manually initiate the replication of a DCE LFS fileset by using Release Replication with the fileset, or you can automate the process by using Scheduled Replication with the fileset. With Release Replication, you issue a command that updates the read-only copies of a read/write fileset whenever you want to release new copies of the fileset. This type of replication is useful for filesets whose replication you want to control closely. With Scheduled Replication, you specify parameters that control how often read-only copies are to be updated. The Replication Server then updates the copies according to the specified intervals. This type of replication is useful for filesets whose replication can be performed asynchronously.

Mounting Filesets

To make the contents of a DCE LFS or non-LFS fileset visible and accessible to users in the DCE namespace, you must attach the fileset to the namespace through a mount point. In DFS, you use the **fts crmount** command to create a mount point for a fileset. A fileset is mounted automatically once a mount point is created for it, so you do not have to issue additional commands to attach the fileset. There are several types of mount points; the tasks in this chapter all use regular mount points, which are the most common type.

A DFS mount point appears and functions like a regular directory, but structurally it is a special symbolic link that indicates the name of the fileset associated with the mount point. Each fileset has a directory structure whose root directory has the same name as the fileset's mount point. You can create standard subdirectories within the fileset's root directory. You can also create other mount points in the directory; these mount points look like subdirectories, but they are associated with files in their own filesets rather than with files in the mount-level directory's fileset.

When the Cache Manager traverses a pathname to locate a file that resides in your cell, it begins at the cell's top-level fileset (**root.dfs**). As it traverses the file's pathname, the Cache Manager accesses a different fileset whenever it encounters a mount point.

Most filesets are mounted with regular mount points. When the Cache Manager encounters a regular mount point in a read-only fileset, it attempts to access the read-only version of the fileset named by the mount point. It also attempts to access the read-only version of any fileset whose mount point it encounters further in the file's pathname. However, if a fileset in the pathname is not replicated, the Cache Manager accesses the read/write version of the fileset. From that point on, it continues to access the read/write version of each fileset it encounters in the remainder of the pathname unless it is explicitly directed to access the read-only (or backup) version of a fileset.

Given how the Cache Manager traverses mount points, to be able to access the read-only version of a fileset, you must replicate all filesets mounted at higher levels in the file system hierarchy. In other words, you must create read-only copies of the fileset that contains the mount point for the fileset and all filesets above it in the file system. (See “Using Mount Points” on page 173 for more information about the different types of mount points and how the Cache Manager accesses them.)

Standard Options and Arguments

When using the **fts** commands to create, manipulate, or delete filesets, you can often add the **-verbose** option to receive detailed information from the **fts** program about its actions as it executes the command. This can be particularly helpful if an operation fails for a reason you do not understand. The amount of additional information produced by the **-verbose** option varies for different commands.

The following options and arguments are common to many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the Reference part of this guide and reference for complete details about each command.)

- The **-fileset** *name* option is the complete name (for example, *user.sandy*) or ID number (for example, **0,,34692**) of the fileset to be used in the command.
- The **-server** *machine* option is the File Server machine to be used in the command. Unless otherwise indicated, you can use any of the following to specify the File Server machine:
  - The machine’s DCE pathname (for example, */.../abc.com/hosts/fs1*)
  - The machine’s host name (for example, **fs1.abc.com** or **fs1**)
  - The machine’s IP address (for example, **11.22.33.44**)
- The **-aggregate** *name* option is the device name (for example, **/dev/lv01**), aggregate name (for example, **lfs1** or **/usr**), or aggregate ID (for example, **3** or **12**) of the aggregate or partition to be used in the command. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file.
- The **-cell** *cellname* option specifies the cell with respect to which the command is to be run (for example, *abc.com*). The default is the local cell of the issuer of the command.
- The **-noauth** option directs the **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.
- The **-localauth** option directs the **fts** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server** (for example, */.../abc.com/hosts/fs1/dfs-server*). Do not confuse a machine’s DFS server principal with its unique **self** identity. (See “Preparing for Exporting” on page 142 for information about DFS server principals.)

  Use the **-localauth** option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

The **-cell**, **-noauth**, and **-localauth** options are always optional.

Exporting Aggregates and Partitions

Before exporting a DCE LFS aggregate or a non-LFS partition (non-LFS aggregate) from a File Server machine, you must ensure that an RPC binding exists for the DCE pathname of the machine and that a DFS server principal and account exist for the machine. You must also ensure that a server entry exists for the machine. The RPC binding is created in CDS, the DFS server principal is created in the Registry Database, and the server entry is registered in the FLDB. (See “Preparing for Exporting” on page 142 for a description of these prerequisites and additional requirements for exporting.)

Prior to exporting a DCE LFS aggregate, you must use the **newaggr** command to construct the aggregate from a raw disk partition. This command formats the partition for use as a DCE LFS aggregate. It is similar to the UNIX **newfs** command, which is used to format a partition. You issue it once for each DCE LFS aggregate that you want to create. The partition to be initialized as a DCE LFS aggregate must be neither mounted locally nor exported to the DCE namespace when you issue the **newaggr** command.

Conversely, before exporting a non-LFS partition for use as an aggregate in DFS, you must create the partition and mount it locally using the **newfs** and **mount** commands or their equivalents. You must also create an entry for the partition in the local **fstab** file or its equivalent.

To make data on a DCE LFS aggregate or non-LFS partition available in the DCE namespace, you must issue the **dfsexport** command to export the aggregate or partition. Before using the **dfsexport** command, include an entry in the *dcelocal*/**var/dfs/dfstab** file for each aggregate or partition to be exported. The **dfsexport** command reads the **dfstab** file to determine which aggregates and partitions can be exported. It then exports the indicated devices. You typically add the **dfsexport** command to a machine’s initialization file (**/etc/rc** or its equivalent) to automatically export aggregates and partitions at system startup. Note that the **dfsexport** command will not export an aggregate or partition that is currently exported.

Because a non-LFS partition can store only one fileset, you register that fileset in the FLDB with the **fts crfldbentry** command *before* you export the partition to the namespace. Using the **fts crfldbentry** command, you specify a name to be associated with the fileset; the FL Server allocates a fileset ID number for the new fileset. You use this fileset ID number when you create the entry for the partition in the **dfstab** file. After the partition is exported, you use the **fts crmount** command to create a mount point for the fileset that the partition contains. The **fts crmount** command makes a fileset visible in the DCE namespace.

Conversely, you must use the **dfsexport** command to export a DCE LFS aggregate to the DCE namespace *before* the aggregate can store filesets. Once a DCE LFS aggregate is exported, you can create filesets on it with the **fts create** command, and you can create mount points for the filesets with the **fts crmount** command. You specify a name for a DCE LFS fileset with the **fts create** command; the fileset is automatically assigned an ID number and registered in the FLDB.

Note that files from a non-LFS partition should not be open when you export the partition. DFS grants tokens for files from a non-LFS partition that are accessed after the partition is exported, but it cannot grant tokens for files from the partition that are accessed before the partition is exported. As a result, DFS cannot effectively synchronize file access between users who opened files before the partition was exported and users who open files after the partition is exported because only the latter have tokens.

The following subsections describe these steps and the commands that are used to perform them in more detail. (See the Reference part of this guide and reference for more information about a specific DFS command.)

Preparing for Exporting

Several prerequisites must be met before you can export either a DCE LFS aggregate or a non-LFS partition from a File Server machine. The following server processes must be running before any of the other steps described in this or the following subsections are attempted:

- A CDS server process (**cdsd**) must be running in the cell, and a CDS advertiser process (**cdsadv**) and a CDS clerk process (**cdsclerk**) must be running on the machine; use the appropriate CDS command to verify that the CDS processes are running. Note that the CDS advertiser on a machine automatically spawns a new CDS clerk for each user on the machine.
- A Security Server process (**secd**) must be running in the cell, and a security client process **dced** must be running on the machine; use the appropriate Security Service command to verify that the processes are running.
- The **dced** process must be running on the machine; use the appropriate **dcecp** command to verify that the process is running.
- An FL Server process must be running in the cell. You can use the **bos status** command to verify that the **flserver** process is running, or you can use the **fts lsfldb** command to list information about the **root.dfs** fileset, which must exist, thus verifying that the **flserver** is actually functioning.

In addition, an RPC binding must exist for the DCE pathname of the File Server machine in CDS, a corresponding DFS server principal must exist for the machine in the Registry Database, and a server entry must exist for the machine in the FLDB. The following subsections describe these topics and additional preparation that must be completed if the machine is to export aggregates or partitions.

Creating an RPC Binding for a File Server Machine

Note: On DCE 2.2 for AIX, this step is performed for you when you configure DFS servers using the **config.dfs** command or the AIX SMIT utility.

A File Server machine must have an RPC binding in CDS for its DCE pathname. The **fts** program uses the RPC binding to contact the File Server machine when it needs to communicate with the **ftserver** process on the machine. This RPC binding includes the server machine's network address, an identifier for the protocol used to communicate with the machine, and an endpoint for communications with the endpoint mapper service of the **dced** process on the machine.

In DCE, server machines are identified by DCE pathnames of the form ***/.../*** *cellname***/hosts/** *hostname*; for example, */.../abc.com/hosts/fs1*. The entry in CDS for the RPC binding defined for each server machine must have a name of the form ***/.../*** *cellname***/hosts/** *hostname***/self**; for example, */.../abc.com/hosts/fs1/self*.

Note that the DFS server principal for the machine is similarly derived. The abbreviated server principal registered for the machine in the FLDB is similar in form to the RPC binding and DFS server principal.

Note: The element that follows the *cellname* in the pathname is not well known; for example, **hosts** could be **dfs-hosts**. However, the string used for the element must be applied consistently to all such names in the cell.

To create an RPC binding for a File Server machine, use the **dcecp** command to create the structure for the RPC binding in CDS. The entry for the RPC binding in CDS must have a name of the form ***/.../*** *cellname***/hosts/** *hostname***/self**.

Creating a DFS Server Principal for a File Server Machine

Note: On DCE 2.2 for AIX, this step is performed for you when you configure DFS servers using the **config.dfs** command or the AIX SMIT utility.

A File Server machine must also have a DFS server principal and associated account in the local Registry Database. The DFS server principal name is used to establish an authenticated connection to the DFS server machine.

In DCE, server machines are identified by DCE pathnames of the form ***/.../*** *cellname***/hosts/** *hostname*; for example, */.../abc.com/hosts/fs1*. The DFS server principal is of the form ***/.../*** *cellname***/hosts/** *hostname***/dfs-server**; for example, */.../abc.com/hosts/fs1/dfs-server*. A machine's DFS server principal is similar in appearance to the name of its RPC binding, the difference being that the last element of the RPC binding name is **self**, whereas the corresponding element of the DFS server principal is **dfs-server**. The two elements also differ in that the RPC binding is defined in CDS, while the DFS server principal is registered in the Registry Database. Note again that **hosts** is not a well-known element of the name.

An abbreviation of the DFS server principal registered in the Registry Database must be used as the principal name associated with the machine's entry in the FLDB. Continuing with the previous example, **hosts/fs1** is the abbreviated DFS server principal associated with the FLDB entry for the machine whose DFS server principal in the Registry Database is */.../abc.com/hosts/fs1/dfs-server*. (The full DFS principal name of a server machine is also associated with a server encryption key in a keytab file; see "Chapter 4. Using Administrative Lists and Keytab Files" on page 95 for more information on server encryption keys.)

Use the **dcecp principal create** command to create a DFS server principal and associated account in the Registry Database for the File Server machine from which aggregates and partitions are to be exported. The DFS server principal must be of the form ***/.../****cellname***/hosts/***hostname***/dfs-server**.

Creating a Server Entry for a File Server Machine

Note: On DCE 2.2 for AIX, this step is performed for you when you configure DFS servers using the **config.dfs** command or the AIX SMIT utility.

Before it can house an exported aggregate or partition, a File Server machine must also have a server entry in the FLDB. The **fts crserverentry** command is used to register a File Server machine's server entry in the FLDB. The server entry stores information about the machine, such as its network addresses (a server entry can store up to four network addresses), its abbreviated DFS server principal name, the

number of fileset entries in the FLDB that can be associated with it, and the group of administrators that ″owns″ (possesses special administrative privileges for) the server entry.

The **-principal** option of the **fts crserverentry** command is used to specify the abbreviated DFS server principal to be registered with a machine's server entry. The abbreviated DFS server principal is of the form **hosts/***hostname*. For example, the DFS server principal */.../abc.com/hosts/fs1/dfs-server* would be abbreviated to **hosts/fs1** for use with the machine's server entry in the FLDB.

The **-quota** option of the **fts crserverentry** command is used to limit the number of filesets (read/write, read-only, and backup) that can reside on a File Server machine. The entry for each fileset in the FLDB defines the File Server machine on which each version of the fileset resides. Each server entry records the total number of filesets that are listed in fileset entries as residing on the File Server machine. No more than the number of filesets that are specified with the **-quota** option can be recorded in the FLDB as residing on the machine at any given time.

The **-owner** option of the command is used to specify the group that owns the server entry. Members of this group can administer the FLDB entries for all filesets on the File Server machine. The administrators in the group need not be included on the **admin.fl** list for the entire cell, which would allow them to modify all of the fileset entries in the FLDB in that cell. The same group can be given ownership of the server entries for all of the File Server machines in an administrative domain, which is a collection of File Server machines administered by the same system administrators. Members of the group can then manipulate the FLDB entries for all of the filesets in the domain. A foreign group cannot own a server entry.

The following additional commands are also provided for the manipulation of server entries in the FLDB:

- The **fts lsserverentry** command lets you list current server entries. In a multihomed server environment (where servers have more than one connection), the command lists all of the machine specifications (host names or IP addresses) currently known for that server.
- The **fts edserverentry** command allows you to edit an existing preference entry. If a server has more than one connection, the additional addresses must be entered through this command. The **fts crserverentry** command can only specify one connection for a server. You can remove connections from a server entry with the **fts edserverentry** command.
- The **fts delserverentry** command lets you remove an existing server entry.

The following subsections provide information about the various server entry manipulation commands.

Creating a Server Entry for a Machine: To create a server entry for a File Server machine, do the following:

- Verify that you have the necessary privilege. You must be included in the **admin.fl** file on each Fileset Database machine. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
- Use the **fts crserverentry** command to create a server entry in the FLDB for the machine:

```
$ fts crserverentry -server machine -principal name \
 [-quota entries] [-ownergroup]
```

The **-server** *machine* option specifies the DCE pathname, host name, or IP address of the server machine whose entry is to be added to the FLDB. The command fails if a network address in use by another server entry is specified with this option.

The **-principal** *name* option is the abbreviated DFS server principal name of the machine to be registered in the FLDB (for example, **hosts/** *hostname*). The machine's principal name in the Registry Database must match this name.

The **-quota** *entries* option sets a limit on the number of fileset entries (read/write, read-only, and backup) in the FLDB that can be associated with the server. If this option is omitted, the default is 0 (zero), meaning that an unlimited number of fileset entries can be associated with the server.

The **-owner** *group* option specifies the name of the group that is the owner of the server entry. A group can be specified by a full or abbreviated group name (for example, **/.../** *cellname***/** *group_name* or just *group_name*). Foreign groups cannot own a local server entry. If this option is omitted, no group owns the server entry; the value **<nil>** is reported as the owner.

Listing Server Entries for Machines: Use the **fts lsserverentry** command to list either the server entry for a specific File Server machine or all current server entries from the FLDB:

```
fts lsserverentry {-server  machine | -all }
```

The **-server** *machine* option specifies the DCE pathname, host name, or IP address of the server machine whose entry in the FLDB is to be displayed. Use this option or use the **-all** option.

The **-all** option specifies that the entries for all server machines in the FLDB are to be displayed. Use this option or use the **-server** option.

Editing a Server Entry for a Machine: To edit the server entry for a File Server machine, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.fl** file on each Fileset Database machine. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Use the **fts edserverentry** command to modify any aspect of an existing server entry in the FLDB. For example, the command can be used to add an additional network address to an existing entry for a machine.

   ```
   $ fts edserverentry -server machine \
    [{-rmaddr  | -addaddr address | -changeaddr address}] \
    [-principal name] [-quota entries] [{-owner   group \
   | -noowner}]
   ```

   The **-server** *machine* option specifies the DCE pathname, host name, or IP address of the server machine whose entry in the FLDB is to be modified. Specify the network address if the **-rmaddr**, **-addaddr**, or **-changeaddr** option is used with the command.

   The **-rmaddr** option removes the network address specified with **-server** from the FLDB. The command fails if the specified address is the only address present for the machine in the FLDB. If you use this option, do not use the **-addaddr** or **-changeaddr** option.

The **-addaddr** *address* option adds the additional address specified with this option to the FLDB for the machine specified with **-server**. A machine can have up to four addresses associated with its entry in the FLDB. If you use this option, do not use the **-rmaddr** or **-changeaddr** option.

The **-changeaddr** *address* option changes the address in the FLDB specified with **-server** to the address specified with this option. If you use this option, do not use the **-rmaddr** or **-addaddr** option.

The **-principal** *name* option changes the abbreviated DFS server principal name of the machine registered in the FLDB (for example, **hosts/** *hostname*). The machine's principal name in the Registry Database must match this name. Omit this option to leave the DFS server principal registered for the machine unchanged.

The **-quota** *entries* option changes the limit on the number of fileset entries (read/write, read-only, and backup) in the FLDB that can be associated with the server. Omit this option to leave the quota for the number of fileset entries unchanged.

The **-owner** *group* option changes the group that is the owner of the server entry. In the entry, the specified group replaces the current owning group, if there is any. A group can be specified by a full or abbreviated group name (for example, ***/.../*** *cellname****/*** *group_name* or just *group_name*). Foreign groups cannot own a local server entry. Use this option or use the **-noowner** option; omit both options to leave the current owning group unchanged.

The **-noowner** option specifies that no group is to own the server entry. In the entry, the empty group ID, which is displayed as **<nil>**, replaces the group that currently owns the server entry; the entry is unchanged in this regard if no group presently owns the server entry. Use this option or use the **-owner** option; omit both options to leave the current owning group unchanged.

Deleting a Server Entry for a Machine:

Note: On DCE 2.2 for AIX, this step is performed for you when you unconfigure DFS servers using the **unconfig.dfs** command or the AIX SMIT utility.

To remove the server entry for a File Server machine, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.fl** file on each Fileset Database machine. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Use the **fts delserverentry** command to delete the server entry for a machine from the FLDB. The command fails if any fileset entry in the FLDB references the server as the location of the fileset.
3.

   ```
   $ fts delserverentry -server machine
   ```

   The **-server** *machine* option specifies the DCE pathname, host name, or IP address of the server machine whose entry in the FLDB is to be removed.

Preparing a File Server Machine for Exporting

Note: On DCE 2.2 for AIX, this step is performed for you when you configure DFS servers using the **config.dfs** command or the AIX SMIT utility.

The following additional prerequisites must be met before a File Server machine can begin to export aggregates or partitions:

- The BOS Server (**bosserver** process) must be running on the machine.
- A keytab file and a server encryption key must exist on the machine.
- The **dfsbind** process must be running on the machine.
- The **fxd** process must be running on the machine.
- The Fileset Server (**ftserver** process) must be running on the machine.
- The Replication Server (**repserver** process) must be running on the machine, if the machine is to house read-only DCE LFS filesets.

The following procedure provides instructions for starting these processes and generating a key. The instructions assume that the **dcecp keytab create** command has already been used to create a keytab file on the machine.

1. Log in as **root** on the machine.
2. Start the BOS Server (**bosserver** process) on the machine with the **bosserver** command, using the **-noauth** option to disable DFS authorization checking on the server machine. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for a thorough description of DFS authorization checking.) The process automatically creates the **admin.bos** file when it starts.

   ```
   # bosserver -noauth
   ```

   The **-noauth** option starts the **bosserver** with DFS authorization checking turned off.

3. Use the **bos addadmin** command to add the necessary administrative users and groups to the **admin.bos** file. Make sure you are included in the list of users or groups added to the list. You must use the **-noauth** option to use the identity **nobody** as the identity of the issuer of the command.

   ```
   # bos addadmin -server machine -adminlist admin.bos \
    [-principal name...] [-group name ...] -noauth
   ```

   The **-adminlist** *admin.bos* option specifies that principals and groups are to be added to the **admin.bos** list on the machine indicated with the **-server** option.

   The **-principal** *name* option specifies the principal name of each user to be added to the **admin.bos** list. A user from the local cell can be specified by a full or an abbreviated principal name (for example, ***/.../ cellname/ username*** or just *username*); a user from a foreign cell can be specified only by a full principal name.

   The **-group** *name* option specifies the name of each group to be added to the **admin.bos** list. A group from the local cell can be specified by a full or an abbreviated group name (for example, ***/.../ cellname/ group_name*** or just *group_name*); a group from a foreign cell can be specified only by a full group name.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

4. Add a server encryption key to the keytab file on the machine with the **bos genkey** command, again using the **-noauth** option. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for complete details about managing a keytab file.)

   ```
   # bos genkey -server machine -kvno version_number -noauth
   ```

The **-kvno** *version_number* option is the key version number of the new key. Valid arguments for this option are decimal integers from 0 (zero) to 255.

The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

5. Enable DFS authorization checking on the machine with the **bos setauth** command, once again using the **-noauth** option.

   ```
   # bos setauth -server machine -authchecking on -noauth
   ```

   The **-authchecking on** option enables DFS authorization checking by removing the **NoAuth** file from the machine specified with the **-server** option.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

6. Start the **dfsbind** process on the machine.

   ```
   # dfsbind
   ```

7. Start the **fxd** process to initialize the File Exporter in the kernel of the machine. Specify the name of the proper administrative group with the **-admingroup** option. (See “Chapter 3. Using ACLs and Groups” on page 63 for more information about using administrative groups.)

   ```
   # fxd -admingroup group
   ```

   The **-admingroup** *group* option specifies the group that can administer the File Exporter on the machine. A group from the local cell can be specified by a full or an abbreviated group name (for example, ***/.../ cellname/ group_name*** or just *group_name*); a group from a foreign cell can be specified only by a full group name. (You may add any other applicable options; see the *Transarc DCE DFS Administration Reference* for complete information about the **fxdprocess).**

8. Log out as **root** from the machine to return to your authenticated DCE identity.
9. Start the Fileset Server (**ftserver** process) with the **bos create** command. (See “Chapter 5. Monitoring and Controlling Server Processes” on page 111 for complete information about starting a server process.) The **admin.ft** file is created automatically when the process starts.

   ```
   $ bos create -server machine -process  ftserver -type simple \
   -cmd  dcelocal/bin/ftserver
   ```

   The **-server** option names the server machine on which to create the new process. The BOS Server on this machine executes the command. If you want to run this command using a privileged identity, specify the File Server machine using the full DCE pathname. If you want to run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the File Server machine with either the machine’s host name or IP address.

   The **-process** *ftserver* option specifies that the process to be created and started is to be identified by the name **ftserver**.

   The **-type simple** option specifies that the **ftserver** process is to be a **simple** process.

   The **-cmd**/*dcelocal*/**bin**/**ftserver** option provides the full pathname to the binary file for the **ftserver** process.

10. Use the **bos addadmin** command to add the necessary administrative users and groups (and possibly server machines) to the **admin.ft** file.

    ```
    $ bos addadmin -server machine -adminlist admin.ft \
     [-principal name...] [-group name...]
    ```

    The **-server** option names the server machine that houses the administrative list to which principals, groups, or both are to be added. The BOS Server on this machine executes the command. If you want to run this command using a privileged identity, specify the File Server machine using the full DCE pathname. If you want to run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the File Server machine with either the machine's host name or IP address.

    The **-adminlist** *admin.ft* option specifies that principals and groups are to be added to the **admin.ft** list on the machine indicated with the **-server** option.

    The **-principal** *name* option specifies the principal name of each user or server machine to be added to the **admin.ft** list. A principal from the local cell can be specified by a full or an abbreviated principal name (for example, ***/.../cellname/username*** or just *username*); a principal from a foreign cell can be specified only by a full principal name.

    The **-group** *name* option specifies the name of each group to be added to the **admin.ft** list. A group from the local cell can be specified by a full or an abbreviated group name (for example, ***/.../cellname/group_name*** or just *group_name*); a group from a foreign cell can be specified only by a full group name.

11. Start the Replication Server (**repserver** process) with the **bos create** command. No administrative list is associated with the **repserver** process.

    ```
    $ bos create -server machine -process repserver -type simple \
    -cmd dcelocal/bin/repserver
    ```

    The **-server** option names the server machine on which to create the new process. The BOS Server on this machine executes the command. If you want to run this command using a privileged identity, specify the File Server machine using the full DCE pathname. If you want to run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the File Server machine with either the machine's host name or IP address.

    The **-process** *repserver* option specifies that the process to be created and started is to be identified by the name **repserver**.

    The **-type simple** option specifies that the **repserver** process is to be a **simple** process.

    The **-cmd**/*dcelocal*/**bin**/**repserver** option provides the full pathname to the binary file for the **repserver** process.

12. After the Fileset Server process is started, use the **fts statftserver** command to verify that the process is performing requested actions. This command is useful mainly if you believe the process is not functioning properly.

    ```
    $ fts statftserver -server machine
    ```

The **fts statftserver** command displays the message **No active transactions on** *machine* if the Fileset Server is functioning properly. It displays additional information if the Fileset Server is currently performing an action. Depending on the information displayed, the Fileset Server may or may not be functioning properly.

Exporting DCE LFS Aggregates

The following subsections introduce and describe the steps that are involved in initializing and exporting a DCE LFS aggregate. Before exporting a DCE LFS aggregate to the DCE namespace, the prerequisites described in the previous section must be met: the necessary Directory Service, Security Service, RPC, and DFS server processes must be running; an RPC binding must exist for the DCE pathname of the machine; a DFS server principal must exist for the machine; a server entry must exist for the machine; and a keytab file and a key must exist on the machine. You must also initialize the aggregate by formatting it with the **newaggr** command before it can be exported.

An Overview of Initializing DCE LFS Aggregates

Prior to creating a DCE LFS aggregate, use the **newaggr** command to initialize the raw partition on which the aggregate is to reside by formatting it for use as a DCE LFS aggregate. The **newaggr** command creates the metadata structure used by the DCE LFS for ACL support, logging, multiple fileset storage, and other fileset-related operations. It also allocates temporary space for use by the DCE LFS log for faster restarts after system failures. The DCE LFS log is not a file; it is a structure that resides on an aggregate. If the system fails, the logged metadata that was written to disk is replayed at system restart to return the system to a consistent state.

Because the **newaggr** command overwrites all data on the partition being initialized, the partition being initialized should not contain data you want to save when the command is issued. Also, the command fails if the partition or aggregate being initialized is currently exported to the DCE namespace. It also fails if the aggregate to be initialized houses a locally mounted fileset. Finally, if the partition is mounted locally, the **newaggr** command causes the kernel to panic. (Note that a non-LFS partition must be mounted locally before it can be exported.)

If you are uncertain about which arguments to supply with the **newaggr** command, execute the command with the **-noaction** option. This option directs the command to report on what it would do without actually modifying the partition. When using the **-noaction** option, supply the other options as you would when actually executing the command.

Note that DCE LFS reserves a variable amount of disk space on every DCE LFS aggregate. By default, DCE LFS reserves 2 megabytes of disk space on an aggregate, but it never reserves less than 1% or more than 10% of the total size of an aggregate (for example, it reserves only 1.5 megabytes on an aggregate whose total size is only 15 megabytes). DCE LFS reserves the disk space for internal purposes (for example, to avoid potential problems with routine administrative operations such as fileset moves and clones). The reserved space is not directly accessible to users and administrators.

In operating systems that support logical volumes, the **newaggr** command can be used to initialize a logical volume as a DCE LFS aggregate. In such cases, all of the command's functionality described here with respect to a disk partition applies to the logical volume.

Note: On AIX, you should create an AIX logical volume and then run the **newaggr** command to initialize the logical volume as a DCE LFS aggregate. It is useful to set the logical volume type to **lfs** when creating it (the default is **jfs**) to make it easier to identify it as a DCE LFS aggregate. It is also useful to name the AIX logical volumes with a **lvlfs** prefix (for example, **/dev/lvlfs01**).

Initializing a DCE LFS Aggregate

CAUTION:

Do not use the newaggr command to initialize a non-LFS partition, especially if it contains data you want to retain, unless you want to convert the partition to a DCE LFS aggregate; the command destroys all data on the specified partition. Also, do not use the command on a locally mounted partition; doing so causes the kernel to panic. Finally, do not use the command on a partition or aggregate that is currently exported to the DCE namespace or on an aggregate that houses a locally mounted fileset; the command fails in these cases.

To initialize a DCE LFS aggregate, do the following:

1. Log in as **root** on the machine on which the new aggregate is to be initialized.
2. Issue the **newaggr** command to initialize the aggregate. (See the Reference part of this guide and reference for more detailed information about the **newaggr** command.)

   ```
   # newaggr -aggregate name -blocksize bytes -fragsize bytes \
   [-initialempty blocks] [-aggrsize  blocks] [-logsize  blocks] \
   [-overwrite ] [-verbose ] [-noaction ]
   ```

   The **-aggregate** *name* option is the device name or aggregate name of the disk partition to be initialized as a DCE LFS aggregate. These identifiers are specified in the first and second fields of the entry for the aggregate in the *dcelocal***/var/dfs/dfstab** file.

   The **-blocksize** *bytes* option is the number of bytes to be available in each DCE LFS block on the aggregate. Allowable values are the powers of 2 from 1024 to 65,536.

   The **-fragsize** *bytes* option is the number of bytes to be available in each DCE LFS fragment on the aggregate. Allowable values are the powers of 2 from 1024 to the number of bytes specified with the **-blocksize** option.

   The **-initialempty** *blocks* option is the number of DCE LFS blocks to be left empty at the beginning of the partition when the aggregate is initialized. Allowable values are from 0 (zero) to 65,536 divided by the number of bytes specified with the **-blocksize** option; for example, if 65,536 is specified with the **-blocksize** option, the **-initialempty** option can be 0 or 1. If this option is omitted, one block is left empty.

   The **-aggrsize** *blocks* option is the total number of DCE LFS blocks that are to be available on the aggregate. Because this value cannot exceed the size of the partition, it can be used only to restrict the size of the aggregate. It must be

large enough to accommodate at least the log and any blocks left empty at the beginning of the partition. If this option is omitted, the size of the partition being initialized is used.

The **-logsize** *blocks* option is the number of DCE LFS blocks to be reserved for the log on the aggregate. This value cannot exceed the number of DCE LFS blocks used for the **-aggrsize** option, and it must specify at least enough blocks for the log to be initially created. If this option is omitted, 1% of the total number of DCE LFS blocks on the aggregate (-aggrsize) is used.

The **—overwrite** option specifies that an existing file system found on the partition can be overwritten. If this option is omitted and a file system is found on the partition, the command informs you that a file system already exists. It then terminates with an exit code of at least 16 without overwriting the existing file system.

The **-noaction** option directs the command to display information about what it would do without actually modifying the partition. Include the other options as you would to actually execute the command. The command displays the default values it would use for its options and informs you if the partition already contains a file system.

Exporting a DCE LFS Aggregate

To export a DCE LFS aggregate, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine from which the aggregate is to be exported, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine from which the aggregate is to be exported. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directories in which the mount points for any filesets are to be created. If necessary, issue the **dcecp acl show** command to check the ACL permissions for the directories. Note that you need to have the **w** and **x** permissions for any directories in non-LFS filesets.
3. Log in as **root** on the machine from which the aggregate is to be exported.

   Note: On DCE 2.2 for AIX, the following steps can be combined into one step by using the **mkfilesys.dfs** command or the AIX SMIT utility.
4. Use a text editor to edit the **dfstab** file to include an entry for the DCE LFS aggregate to be exported. The following fields appear for each entry in the file, in the order listed. Each field must be separated by a minimum of one space or tab; each entry must be on a separate line.
   - Device Name: The block device name of the aggregate (for example, **/dev/lv03**).
   - Aggregate Name: The name to be associated with the exported aggregate. An aggregate name can contain any characters, but it can be no longer than 31 characters, and it must be different from any other aggregate name in the file. Aggregate names cannot be abbreviated, so you should choose a short, explicit name (for example, **lfs1**).
   - File System Type: The identifier for the file system type of the aggregate. For DCE LFS aggregates, this must be **lfs**. It must be in lowercase letters.

- Aggregate ID: A positive integer to act as the aggregate ID of the exported aggregate. The integer must be different from any other aggregate ID in the **dfstab** file. (If the ID is changed after the aggregate contains one or more filesets, fileset operations on those filesets will fail.)

The following entry from a **dfstab** file is for a DCE LFS aggregate:

```
/dev/lv03  lfs1  lfs  3
```

5. Issue the **dfsexport** command to export the aggregate to the DCE namespace. Before exporting, this command reads the **dfstab** file to determine which aggregates and partitions are available to be exported. Omit all of the command's options to list the aggregates and partitions currently exported from the local disk to the DCE namespace.

   # dfsexport [{-all | -aggregate *name***}] [-type** *name*]

   The **-all** option specifies that all aggregates and partitions listed in the **dfstab** file are to be exported. Use the **-all** option with the **-type** option to export only DCE LFS aggregates or only non-LFS partitions. Use this option or use the **-aggregate** option.

   The **-aggregate** *name* option specifies the device name or aggregate name of a specific aggregate or partition. These names are specified in the first and second fields of the entry for the aggregate or partition in the **dfstab** file. Use the **-aggregate** option or use the **-all** option.

   The **-type** *name* option is the file system type to be exported. Specify **lfs** to export only DCE LFS aggregates; specify **ufs** to export only non-LFS partitions. Use the **-type** option only with the **-all** option (it is ignored if it is used without **-all**); omit **-type** and use **-all** to export all aggregates and partitions.
6. Log out as **root** from the machine to return to your authenticated DCE identity.
7. Issue the **fts create** command to create a DCE LFS fileset on the aggregate and register the fileset in the FLDB. The FL Server allocates a unique fileset ID number for the fileset. (See "Creating Read/Write DCE LFS Filesets" on page 158 for detailed information about DCE LFS fileset creation.)

   $ fts create -ftname *name* **-server** *machine* **-aggregate** *name*

   The **-ftname** *name* option is the complete name to be associated with the fileset being created. The name can contain no more than 102 characters, and it must contain at least one alphabetic character or an _ (underscore). (See "Fileset Names" on page 135 for more information on fileset naming conventions.)

   Repeat the **fts create** command for each fileset you want to create on the aggregate.
8. Enter the **fts crmount** command to create a mount point in the file system for the new DCE LFS fileset. This makes the contents of the fileset visible to other users. (See "Using Mount Points" on page 173 for more information about mounting filesets.)

   $**fts crmount -dir** *directory_name* **-fileset** { *name* | *ID*}

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

   Repeat the **fts crmount** command for each fileset created in the previous step.

Exporting Non-LFS Partitions

This section describes the steps involved in exporting a non-LFS partition; after it is exported, a non-LFS partition can be referred to as a non-LFS aggregate. Before exporting a non-LFS partition to the DCE namespace, the prerequisites described in “Preparing for Exporting” on page 142 must be met: the necessary CDS, Security Service, RPC, and DFS server processes must be running; an RPC binding must exist for the DCE pathname of the machine; a DFS server principal and account must exist for the machine; a server entry must exist for the machine; and a keytab file and a key must exist on the machine. You must also have created and locally mounted the partition (using the **newfs** and **mount** commands or their equivalents), and you must have listed the partition's name in the local **fstab** file (or its equivalent).

Note: For the DCE 2.2 for AIX implementation, *non-LFS* partitions refers to AIX Journaled File Systems (JFS) or AIX CD-ROM File Systems. No file system activity should be occurring on a JFS file system that is being exported to the DFS filespace. Otherwise, an inconsistent DFS token state can occur. When you configure the DFS File Server to export JFS filesystems, you must start DFS at system start time to ensure that the DFS token state is correct. For information about starting DCE and DFS at system start time, refer to *IBM DCE for AIX, Version 2.2: Quick Beginnings*.

Before exporting a non-LFS partition, make sure that no users have files open on the partition. DFS cannot effectively synchronize file access between users who opened files from a non-LFS partition before the partition was exported and users who open files from the partition after the partition is exported because only the latter have tokens.

To export a non-LFS partition, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine from which the partition is to be exported. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the mount point for the fileset is to be created. If necessary, issue the **dcecp acl show** command to check the ACL permissions for the directory. Note that you need to have the **w** and **x** permissions if the directory is in a non-LFS fileset.

   Note: On DCE 2.2 for AIX, the following steps can be combined into one step by using the **mkfilesys.dfs** command or the AIX SMIT utility.

   CAUTION:

   To ensure proper operation, do not change the CD in the CD-ROM device while the CD-ROM File System is being exported by DFS.

   a. **Before you change CDs, unexport the CD-ROM File System from DFS and unmount it from the local AIX File System.**
   b. **After you place a new CD in the CD-ROM device, locally mount the CD-ROM File System and re-export it to DFS.**
   c. **Executing the cm checkfilesets command at DFS client machines causes the new data to be recognized immediately.**
3. Provide a name for the fileset (non-LFS file system) on the partition to be exported and register the fileset in the FLDB with the **fts crfldbentry** command.

The number specified with the **-aggrid** option is also used as the partition's aggregate ID in the **dfstab** file; it must not already be in use in the **dfstab** file.

The FL Server allocates a unique fileset ID number for the partition's lone non-LFS fileset. The **fts crfldbentry** command returns this ID number, along with two additional ID numbers allocated for read-only and backup versions of the fileset, even though a non-LFS fileset cannot have these versions. Use the read/write ID number returned by the command as the fileset ID in the **dfstab** file.

```
$ fts crfldbentry -ftname name -server machine -aggrid ID
```

The **-ftname** *name* option is the complete name to be associated with the fileset being registered. The name can contain no more than 102 characters, and it must contain at least one alphabetic character or an _ (underscore). (See "Fileset Names" on page 135 for more information on fileset naming conventions.)

The **-aggrid** *ID* option is a positive integer to serve as the aggregate ID for the partition to be exported. The number must not already be in use in the **dfstab** file on the machine where the partition resides.

4. Log in as **root** on the machine from which the partition is to be exported.
5. Use a text editor to edit the **dfstab** file to include an entry for the non-LFS partition to be exported. The following fields appear for each entry in the file, in the order listed. Each field must be separated by a minimum of one space or tab; each entry must be on a separate line. Note that, because a non-LFS partition can contain only a single fileset, you include the fileset ID number with the partition's entry in the **dfstab** file.
   - Device Name: The block device name of the partition; for example, **/dev/lv02**.
   - Aggregate Name: The name to be associated with the exported partition. The aggregate name of a non-LFS partition must match the name of its local mount point (for example, **/usr**). An aggregate name can contain any characters, but it can be no longer than 31 characters, and it must be different from any other aggregate name in the file. Aggregate names cannot be abbreviated, so you should choose a short, explicit name.
   - File System Type: The identifier for the file system type of the partition. For non-LFS file systems, this must be **ufs**. It must be in lowercase letters.
   - Aggregate ID: A positive integer to serve as the aggregate ID of the exported partition. The integer must match the aggregate ID specified with the **-aggrid** option of the **fts crfldbentry** command, and it must be different from any other aggregate ID in the **dfstab** file. (If the ID is changed, fileset operations on the partition's fileset will fail.)
   - Fileset ID: The unique fileset ID number returned by the **fts crfldbentry** command for the fileset on the partition (for example, **0,,18756**). Use the read/write ID number, not the read-only or backup ID number, returned by the command as the value for this field.

   The following entry from a **dfstab** file is for a non-LFS partition:

   ```
   /dev/lv02  /usr  ufs  1  0,,18756
   ```
6. Issue the **dfsexport** command to export the partition to the DCE namespace. Before exporting, this command reads the **dfstab** file to determine which aggregates and partitions are available to be exported. Omit all of the command's options to list the aggregates and partitions currently exported from the local disk to the DCE namespace.

   ```
   # dfsexport [{-all | -aggregate name}] [-type name]
   ```

The **-all** option specifies that all aggregates and partitions listed in the **dfstab** file are to be exported. Use the **-all** option with the **-type** option to export only DCE LFS aggregates or only non-LFS partitions. Use this option or use the **-aggregate** option.

The **-aggregate** *name* option specifies the device name or aggregate name of a specific aggregate or partition. These names are specified in the first and second fields of the entry for the aggregate or partition in the **dfstab** file. Use the **-aggregate** option or use the **-all** option.

The **-type** *name* option is the file system type to be exported. Specify **lfs** to export only DCE LFS aggregates; specify **ufs** to export only non-LFS partitions. Use the **-type** option only with the **-all** option (it is ignored if it is used without **-all**); omit **-type** and use **-all** to export all aggregates and partitions.

7. Log out as **root** from the machine to return to your authenticated DCE identity.
8. Enter the **fts crmount** command to create a mount point in the file system for the new non-LFS fileset. This makes the contents of the fileset visible to other users.

   $ fts crmount -dir *directory_name* **-fileset** {*name* | *ID*}

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

Exporting Aggregates and Partitions at System Startup

Note: With DCE 2.2 for AIX, start all daemons for configured DFS components with the **start.dfs** command. Refer to *IBM DCE for AIX, Version 2.2: Quick Beginnings* for information about configuring DFS and about starting up DCE at system startup on AIX.

To export DCE LFS or non-LFS aggregates at system startup, use a text editor to edit the appropriate initialization file for the File Server machine to include the following DFS commands:

- The **dfsbind** command.
- The **dfsexport** command with the **-all** option to export all partitions and aggregates with entries in the **dfstab** file.
- The **bosserver** command to start the BOS Server. If the *dcelocal***/var/dfs/BosConfig** file includes the recommended entries, this also starts the Fileset Server and the Replication Server on the machine.
- The **fxd** command to start the File Exporter.

These commands may be included in the file **start.dfs** (or its equivalent), which is installed with DFS on the local machine. The file may be automatically modified to start the appropriate processes as you alter your DFS configuration.

Removing Aggregates and Partitions from the Namespace

If you exported a DCE LFS aggregate or non-LFS partition (either by adding the **dfsexport** command to the proper initialization file—**/etc/rc** or its equivalent—or by issuing the command directly), you can issue the **dfsexport** command with the **-detach** option to remove the aggregate or partition from the DCE namespace.

Before the command detaches an exported aggregate or partition, it first revokes all of the tokens for data on the aggregate or partition. When its tokens are revoked, a client flushes the data that is cached from the aggregate or partition, writing any modified data back to the File Server machine. The command does not perform the detach operation if it cannot revoke all necessary tokens; it instead reports that the device is busy. In this case, the command's **-force** option can be included to force completion of the detach operation even if all necessary tokens cannot be revoked.

In general, avoid detaching an aggregate or partition if users are still accessing filesets that reside on it. The **dfsexport** command revokes all tokens for data on the aggregate or partition before it detaches it, but users accessing data from the aggregate or partition will not be able to save the data. Be especially cautious when using the **-force** option, which forces an aggregate or partition to be detached even if all tokens cannot be revoked.

Using DCE LFS Filesets Locally

Mounting a DCE LFS fileset locally does not improve access time to data in the fileset if the fileset is accessed via a DCE pathname. However, access time to data in the fileset is improved if the fileset is accessed via a local pathname. Availability of the fileset is also generally improved because the fileset can still be accessed via a local pathname in the event of a network outage. If data in a DCE LFS fileset that is physically located on the local disk of your machine is available only through the DCE namespace, a network outage makes it impossible to access the data.

Provided the DCE LFS aggregate that contains a fileset is exported and the fileset is mounted in the DCE namespace, you can also access a locally mounted DCE LFS fileset globally. The pathname associated with the fileset is different for local and global access. For example, a fileset that is accessed in the local operating system as */usr/jlw* may be accessed in the DCE namespace as */.../abc.com/fs/usr/jlw*.

Note: A DCE LFS fileset that is mounted locally cannot be moved to a different File Server machine (with the **fts move** command), and it cannot be deleted (with the **fts delete** or **fts zap** command); you must unmount it locally before attempting any of these operations. Also, an aggregate that houses a locally mounted fileset cannot be recovered or salvaged with the **salvage** command, nor can it be reinitialized with the **newaggr** command.

The following instructions describe the steps involved in mounting a DCE LFS fileset locally. It is assumed that the aggregate that houses the fileset has already been initialized with the **newaggr** command and that the fileset to be made available locally has already been created with the **fts create** command.

1. Log in as **root** on the machine. For example, issue the following command:

   ```
   $ su root
   Password: root_password
   ```

2. Create an empty directory on the local machine to serve as the local mount point for the fileset. For example, issue the following command:

   ```
   # mkdir directory_name
   ```

3. Locally mount the DCE LFS using the AIX **mount** command fileset.

   To locally mount a DCE LFS fileset over a local directory mount point, use the following form of the AIX **mount** command:

   ```
   # mount -v lfs -o aggregate=Aggregate_name-n Node Fsname Directory
   ```

   - The **-v***lfs* option specifies that the vfs type is DCE LFS.

- The **-o aggregate** option specifies the aggregate name (or aggregate device name) of the aggregate on the local node that holds the fileset.
- The **-n** *Node* option specifies the local host name.
- The *Fsname* option specifies the DCE LFS fileset to be mounted.
- The *Directory* option specifies the local mount point.

The **mount** command makes the DCE LFS fileset (specified by *Fsname*) available for use at the mount point (specified by*Directory*). The **-o aggregate** option specifies the DCE LFS aggregate that holds the fileset.

It is necessary to indicate the local host name with the **-n** *Node* parameter. The **mount** command returns without an error if you do not specify the local host name, but the mount does not happen.

Here is an example of mounting the DCE LFS fileset (**fileset_1**), which is in the DCE LFS aggregate (**aggregate_1**) on the `nodeA` node, over the `/localdir` directory.

mount -v *lfs* **-o aggregate**=aggregate_1 **-n** *nodeA* **fileset_1 /localdir**

To unmount the fileset, enter the following command:

```
# unmount /localdir
```

Creating Read/Write DCE LFS Filesets

Read/write DCE LFS filesets are created with the **fts create** command. The **fts create** command is used to create only DCE LFS filesets; non-LFS filesets are created by exporting and mounting non-LFS partitions (as described in “Exporting Non-LFS Partitions” on page 154).

Before creating a read/write DCE LFS fileset, select a site for the fileset. A site is an aggregate on the File Server machine where the fileset is to reside. If necessary, issue the **fts aggrinfo** command to make sure the aggregate you choose has enough space to accommodate the fileset. (See “Chapter 7. Managing Filesets” on page 179 for a detailed description of the **fts aggrinfo** command.)

You specify a fileset’s name when you create it with the **fts create** command. A fileset’s name should describe the fileset’s contents; for example, the name *user.terry* describes a fileset that contains data for the user **terry**. A fileset name must also be unique within the local cell. It can be no greater than 102 characters in length, and it must contain at least one alphabetic character or an _ (underscore). (See “Fileset Names” on page 135 for more information on fileset naming conventions.)

The **fts create** command

- Creates a single FLDB entry for the read/write fileset and for any potential read-only and backup versions of the fileset.
- Allocates a fileset ID number for the read/write fileset. It also reserves ID numbers for the read-only and backup versions of the fileset in anticipation of their creation.
- Assigns the name that you specify to the fileset.
- Sets the FLDB site flag for the read/write fileset to **valid**, and sets the site flags for the read-only and backup filesets to **invalid** because they do not yet exist.

- Creates a fileset header at the site File Server machine and aggregate that you designate as the location of the fileset.
- Creates an empty root directory in the fileset. This directory becomes visible when the **fts crmount** command is used to mount the fileset.
- Records null ACLs as the default for use by the root directory of the fileset. Note that, due to the interaction between UNIX mode bits and ACLs, the directory has a set of implicit initial ACLs that grant permissions to different users and groups. (See “Chapter 3. Using ACLs and Groups” on page 63 for information about the interaction between ACLs and UNIX mode bits and for suggestions for initial ACLs.)
- Assigns a default quota of 5000 kilobytes to the fileset.

The following subsections describe the steps involved in creating a read/write DCE LFS fileset. After creating a read/write DCE LFS fileset with the **fts create** command, use the **fts crmount** command to create a mount point for the fileset. The mount point makes the contents of the fileset visible in the DCE namespace. You can use the **fts setquota** command to alter the fileset's default quota of 5000 kilobytes, and you can use the **dcecp acl modify** command to modify the default ACLs of the fileset's root directory.

Once a read/write fileset is created, you can create read-only and backup copies of the read/write fileset. The following sections provide detailed information about creating read-only and backup DCE LFS filesets.

Creating and Mounting a Read/Write Fileset

To create and mount a read/write fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to reside, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset is to reside. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the fileset is to be mounted. If necessary, issue the **dcecp acl show** command to check the permissions for the directory.
3. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate on which the fileset is to be created. (See “Chapter 7. Managing Filesets” on page 179 for a detailed description of the **fts aggrinfo** command.)

   ```
   $ fts aggrinfo -server machine -aggregate name
   ```

   Note: On DCE 2.2 for AIX, the following steps can be combined into one step by using the **mkfilesys.dfs** command or the AIX SMIT utility.
4. Enter the **fts create** command to create the fileset:

   ```
   $ fts create -ftname name -server machine -aggregate name
   ```

   The **-ftname** *name* option is the complete name to be associated with the fileset being created.
5. Enter the **fts crmount** command to create a mount point in the file system for the new fileset. This makes the contents of the fileset visible to other users.

   ```
   $ fts crmount -dir directory_name -fileset { name | ID}
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of

the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

Resetting the Fileset Quota

To reset the quota for the new read/write fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of the list.
2. Issue the **fts setquota** command to change the fileset's quota:

   ```
   $ fts setquota{-path  {filename | directory_name} \
    | -fileset  {name | ID}} -size  kbytes
   ```

   The **-path** *filename* or *directory_name* option is the name of a file or directory on the fileset whose quota you want to set. Use the **-path** option or use the **-fileset** option.

   The **-size** *kbytes* option is the maximum amount of disk space that all of the files and directories in the fileset can occupy, including files referred to by this fileset but housed in another fileset type of this fileset. Specify the value in kilobytes; a value of 1024 kilobytes equals 1 megabyte.

Creating Read-Only DCE LFS Filesets

Replication is the process of creating read-only copies (replicas) of a read/write DCE LFS fileset and placing the copies on multiple File Server machines. Replication increases the availability of the fileset in the event of a network or server outage. If one of the machines that houses the fileset becomes unavailable, the fileset can usually still be accessed from another machine. Replication is not available for non-LFS filesets.

Replicate read/write filesets that match the following criteria:

- The files in the fileset are read much more frequently than they are modified.
- The files in the fileset are heavily used (for example, binary files for text editors or other frequently accessed application programs). Replicating the fileset lets you distribute the load for the files it contains across several machines.
- The files in the fileset must remain available. By replicating the fileset on multiple File Server machines, even if one of the machines that houses the fileset becomes unavailable, the fileset is still available from another machine.
- The fileset is mounted at a high level in the cell's file tree (for example, **root.dfs** and its subdirectories).

The following two types of replication are available for DCE LFS filesets:

- *Release Replication*, which requires you to issue the **fts release** command to explicitly update the read-only versions of the read/write source fileset. The command places a read-only copy of the source fileset on the same File Server machine as the source. The Replication Server (**repserver** process) on each machine that houses a read-only replica then updates the replica on its machine to match the replica stored on the same machine as the read/write fileset. The read-only replicas do not change until you issue the **fts release** command.

  Use Release Replication for a fileset that seldom changes or for a fileset whose replication you need to track closely.

- *Scheduled Replication*, which requires you to specify replication parameters that control the length of time between automatic updates of the read-only replicas. The Replication Server on each machine that houses a replica updates the replica on its machine according to time intervals that you supply. For a fileset that uses Scheduled Replication, you can use the **fts update** command to request an immediate update of the fileset's replicas at any time. The command can be used to update all replicas or only the replica at a specific site.

  Use Scheduled Replication for a fileset if you prefer to have the system automatically update the fileset's replicas and you do not need to monitor the fileset's replication.

A read/write fileset can be replicated via only one of the two types of replication at any one time. The type of replication to be used for a fileset is set or changed with the **fts setrepinfo** command. This command is also used to set the replication parameters to be used with the fileset. Some replication parameters are used with both Release and Scheduled Replication to specify how the replicated data is to be used by Cache Managers; other parameters are used only with Scheduled Replication to define how often Replication Servers are to check for updated versions of the source fileset. (An additional site-specific parameter used with Scheduled Replication is set with the **fts addsite** command; see "Replication Type and Parameters" on page 163 for more information about these parameters.)

Whenever you update a read-only replica by either type of replication, DFS ignores any shared byte-range locks held against the replica. Data accessed by a client during an update is not lost or damaged; however, some of the data sent to the client may change as the replica is updated even though the client holds a shared lock.

Each replica of a read/write fileset resides at a specified replication site (a specific File Server machine and aggregate). Before read-only versions of a fileset can be made, the **fts addsite** command must be used to define the sites where the read-only replicas are to reside. It is not possible to place multiple read-only copies of the same fileset on a single File Server machine, and it is not possible to place a replica of a fileset from one cell on a File Server machine from another cell.

With Release Replication, the **fts addsite** command must be used to define a replication site on the File Server machine on which the source fileset resides before any subsequent replication sites can be added. The site for the replica can be defined on any aggregate on the source's File Server machine. However, it is best to define the site for this replica on the same aggregate as that on which the source fileset resides, in which case the replica is created as a clone of the source fileset. Because it is created as a clone fileset, which has the same structure as a backup fileset, a replica that exists on the same aggregate as the source requires potentially much less space than a full read-only replica created on a different aggregate. (See "Data Sharing Among the Different Types of DCE LFS Filesets" on page 134 for more information about the structure of backup filesets and the sharing of data among the different types of DCE LFS filesets.)

Note: Replication is available in a cell only if the following are true: **root.dfs**, the cell's main read/write fileset, is a DCE LFS fileset; **root.dfs** was mounted with an explicit read/write mount point as a subdirectory of itself (the **root.dfs** fileset) when the cell was configured; and **root.dfs** is replicated. (See "Chapter 2. DFS Configuration Issues" on page 27 for information about configuring **root.dfs** to support replication.)

Replication Information in the FLDB

The FLDB entry for a read/write fileset records the following replication information for the fileset:

- The ID number of the fileset's read-only version
- The fileset's replication type and parameters
- The definitions for the fileset's replication sites

All read-only copies of a read/write fileset share the same fileset ID number in the FLDB. The number, which is reserved when the read/write fileset is created, is one greater than the ID number of the read/write fileset. The fileset ID number of a read-only fileset's source fileset is referred to as the replica's parent ID number.

When a read/write fileset is first created, the status flag for the read-only version in the fileset's FLDB entry is set to **invalid**. When the **fts addsite** command is used to define the fileset's first replication site, the status flag for the read-only version is changed to **valid**. The change is made because it is assumed that replicas of the fileset will be placed at the replication sites shortly after the sites are defined.

With respect to filesets, a site is a specific File Server machine and aggregate on which the fileset resides. The FLDB can record a maximum of 16 sites for all versions of a fileset combined. A fileset's read/write version and backup version (if it exists) share a single site definition. If you define a replication site for a fileset at the same site as its read/write and backup versions, you can then define 15 additional replication sites for the fileset; this approach allows you to define up to 16 replication sites. If you choose not to place a replica of a fileset at the same site as its read/write and backup versions, you can define a maximum of 15 replication sites for the fileset. You should define a replication site for any fileset, especially one that uses Release Replication, at the same site as its read/write fileset.

Preparing for Replication

The following prerequisites must be met before you can replicate a read/write fileset:

- Each File Server machine that is to house a replica of the fileset must have a server entry in the FLDB. (See "Creating a Server Entry for a File Server Machine" on page 143 for information on creating server entries.)
- A Replication Server (**repserver** process) and a Fileset Server (**ftserver** process) must be running on each File Server machine that is to house a replica of the fileset. Use the **bos status** command to verify that the **repserver** and **ftserver** processes are running on the machine at each replication site. (You can also use the **fts statrepserver** command, which is described in "Displaying Replication Status" on page 170, to verify the status of the Replication Server on a machine.)
- You must use the **fts setrepinfo** command to indicate the type of replication and to set the replication parameters to be used with the fileset that is to be replicated. This information is recorded in the FLDB entry for the fileset.
- You must use the **fts addsite** command to add the replication sites to the FLDB entry for the fileset. If necessary, enter the **fts aggrinfo** command to check the available space on a File Server machine's aggregates before defining a replication site on the machine, and use the **fts lsft** command to check the size of the read/write fileset. (See "Chapter 7. Managing Filesets" on page 179 for a

description of the **fts aggrinfo** and **fts lsft** commands.) Remember, a read-only replica must also be stored on the same File Server machine as the read/write fileset with Release Replication.

The following subsections describe how to set the replication type and parameters and how to add replication sites.

Replication Type and Parameters

Before defining any replication sites, you must use the **fts setrepinfo** command to indicate the type of replication to be used and to set the replication parameters for the fileset to be replicated. When initially defining the type of replication to be used with a fileset, include either the **-release** option or the **-scheduled** option with the command to indicate the type of replication to be used.

Most of the options available with the **fts setrepinfo** command determine the replication parameters to be associated with the fileset; the primary exceptions are the **-change** and **-clear** options, which are described later in this section. Replication parameters define the time intervals used by Cache Managers on client machines that are accessing data from the read-only replicas and Replication Servers on File Server machines that house the replicas. Descriptions of the replication parameters follow; each parameter is set with an option of the same name. (Note that the term *replication parameter* is used to refer to the *replication intervals* that are defined for a fileset with the **fts setrepinfo** command, not to the parameters of the command.)

The following parameters apply to *both* Release Replication and Scheduled Replication:

- MaxAge specifies the amount of time the Cache Manager distributes data cached from a read-only replica without attempting to verify that the data is current. The Replication Server maintains information about the currentness of a read-only replica, which it communicates to the Cache Manager via the File Exporter. For Scheduled Replication, a replica must remain current with respect to the read/write source fileset; for Release Replication, a replica must remain current with respect to the read-only replica that resides on the same File Server machine as the read/write source fileset.
- FailAge specifies the amount of time the Cache Manager distributes data cached from a read-only replica if the data cannot be verified as current. The difference between FailAge and MaxAge is the amount of time the Cache Manager continues to distribute data cached from a read-only replica after the data cannot be verified as current. An effective FailAge value is greater than or equal to the MaxAge value.
- ReclaimWait specifies the amount of time the File Exporter waits before it reclaims storage space from deleted files (those not referred to by any directory). It also determines the frequency of the Cache Manager's keep-alive messages to the Replication Server.

  The Cache Manager sends keep-alive messages to indicate that it is still using files from a read-only replica. A file that is being accessed from a replica remains available as long as the Cache Manager continues to notify the Replication Server that the file is still in use and the Replication Server continues to forward these notifications to the File Exporter. This is true even if the file has been removed from all directories on the read/write fileset in the interim. To prevent the File Exporter from reclaiming storage space that was occupied by deleted files, the Cache Manager sends keep-alive messages more frequently than the ReclaimWait interval.

The following parameters apply *only* to Scheduled Replication:

- MinRepDelay specifies how long the Replication Server waits after a read/write fileset changes before it attempts to get a new copy of the fileset. The Replication Server tracks the currentness of replicas by maintaining a whole-fileset token for each fileset. If a Cache Manager changes the read/write fileset, the Replication Server relinquishes its whole-fileset token and waits for at least the time specified by MinRepDelay before requesting a new whole-fileset token.
- MaxSiteAge controls the maximum amount of time that a replica can be out of date. The Replication Server attempts to keep a replica current within this amount of time. A MaxSiteAge value is stored with each site; the MaxSiteAge for a site is set with the **fts addsite** command.
- DefaultSiteAge is the default value to be used as the MaxSiteAge for a replication site if that value is not set with the **fts addsite** command.

Table 6-1 summarizes the six parameters just described. For each parameter, the table describes the command with which the parameter is set, its default value, its usage (if any) with Release Replication, and its usage with Scheduled Replication; usage descriptions include details on the dependencies between the different parameters. Refer to this table when using the **fts setrepinfo** (or **fts addsite**) command to specify the replication parameters for a fileset (or site).

Note: Unless it is absolutely necessary to change them, it is recommended that you use the default parameters (with the exception of MaxAge) that are listed in the table.

Table 8. Descriptions of Replication Parameters

Parameter	**Default**	**Release Replication**	**Scheduled Replication**
MaxAge (`fts setrepinfo`)	2 hours	Required only if FailAge is specified	Required only if FailAge, MinRepDelay, or DefaultSiteAge is specified
FailAge (`fts setrepinfo`)	1 day or twice MaxAge, whichever is larger	Optional	Required only if MinRepDelay or DefaultSiteAge is specified
ReclaimWait (`fts setrepinfo`)	18 hours	Required only if FailAge is specified	Required only if FailAge, MinRepDelay, or DefaultSiteAge is specified
MinRepDelay (`fts setrepinfo`)	5 minutes or one-quarter of DefaultSiteAge, whichever is smaller	Not applicable	Required only if FailAge or DefaultSiteAge is specified
MaxSiteAge (`fts addsite`)	DefaultSiteAge	Not applicable	Required only if DefaultSiteAge is *not* specified

Table 8. Descriptions of Replication Parameters (continued)

Parameter	Default	Release Replication	Scheduled Replication
DefaultSiteAge (`fts setrepinfo`)	One-quarter of MaxAge	Not applicable	Optional

The system uses the guidelines listed in Table 8 on page 164 to calculate default values for each of the parameters *unless* you specify

- FailAge for Release Replication
- FailAge, MinRepDelay, or DefaultSiteAge for Scheduled Replication

Once you specify one of these parameters, the system no longer performs any default calculations; you must specify values for all applicable parameters. The exception is DefaultSiteAge for Scheduled Replication, which is always optional. However, if the other parameters specified with the **fts setrepinfo** command are supplied, the system does not calculate a default value for DefaultSiteAge; you must specify the MaxSiteAge for each replication site with the **fts addsite** command. Also, because the MinRepDelay, MaxSiteAge, and DefaultSiteAge parameters do not apply to Release Replication, they are recorded but otherwise ignored if they are specified for a fileset that uses Release Replication. (They are used if the fileset's style of replication is ever changed to Scheduled Replication.)

Setting Replication Type and Parameters: To set a fileset's replication type and parameters, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

   Note: If you use the command's **-change** option to change a fileset's existing replication type from Release to Scheduled, you must also be included in the **admin.ft** file on the machine on which the read/write fileset resides if a replica actually resides at the replication site on that machine. (The first replication site defined for a fileset that uses Release Replication must be on the same File Server machine as the read/write fileset.)

2. Enter the **fts setrepinfo** command to set the replication parameters for the read/write fileset that is to be replicated. To use the default parameters described in Table 6-1, enter only the fileset name or ID number and the replication type. To specify one or more of the parameters, refer to the table for information on which parameters apply for the two types of replication and the dependencies between the parameters.

   Enter *interval* values as integers with the following abbreviations to designate units: **d** for days, **h** for hours, **m** for minutes, and **s** for seconds. For example, to indicate 3 days and 2 hours, enter **3d2h**. At least one of the four values (days, hours, minutes, or seconds) must be provided, and the unit abbreviations (**d**, **h**, **m**, or **s**) must be used with any integer. The unit abbreviations can be uppercase or lowercase, and they can be entered in any order; for example, **3m2h** is a valid entry for 2 hours and 3 minutes.

   ```
   $ fts setrepinfo -fileset {name | ID} {-release | -scheduled} \
    [-change ] [-maxage interval] [-failage interval] \
    [-reclaimwait interval] [-minrepdelay interval] [-defaultsiteage  interval] \
   [-clear ]
   ```

The **-fileset** *name* or *ID* option is the complete name or ID number of the read/write DCE LFS fileset to be replicated.

The **-release** option specifies that Release Replication is to be used with the fileset. When initially setting the replication type, use this option or use the **-scheduled** option.

The **-scheduled** option specifies that Scheduled Replication is to be used with the fileset. When initially setting the replication type, use this option or use the **-release** option.

The **-change** option is used with **-release** or **-scheduled** to indicate that the replication type is to be changed. When changing the replication type for a fileset, you must include the **-change** option. (See "Changing Replication Type and Parameters".)

The **-maxage** *interval* option is the value for MaxAge. An effective value must be greater than or equal to 2 minutes.

The **-failage** *interval* option is the value for FailAge. An effective value must be greater than or equal to **-maxage**.

The **-reclaimwait** *interval* option is the value for ReclaimWait. An effective value must be greater than 2 hours; do not specify a value less than 90 minutes.

The **-minrepdelay** *interval* option is the value for MinRepDelay. This value must be less than the MaxSiteAge specified for each replication site with the **-maxsiteage** option of the **fts addsite** command.

The **-defaultsiteage** *interval* option is the value for DefaultSiteAge. This value is used as the MaxSiteAge for a replication site if the **-maxsiteage** option is omitted when the **fts addsite** command is used to define the site.

The **-clear** option removes all replication parameters previously defined for the fileset. (See "Changing Replication Type and Parameters".)

Changing Replication Type and Parameters: You can use the **fts setrepinfo** command to change the type of replication or the replication parameters that are associated with a fileset at any time after they are set. Brief descriptions of the operations used to change these values follow:

- *To change some replication parameters*, use the options for the parameters you want to change to indicate the new parameters.
- *To change all replication parameters*, use the **-clear** option to remove all previous replication parameters, and either use the options for the parameters you want to change to indicate the new parameters or omit the options to allow the system to calculate new replication parameters.
- *To change the replication type*, use the **-release** or **-scheduled** option to indicate the new type of replication to be used, and use the **-change** option to indicate that the type is to be changed. Although not required, you may also want to use the **-clear** option to clear all previous replication parameters and then reset them using parameters that are better suited to the new type of replication.

  Because Scheduled Replication imposes more constraints than Release Replication, Release Replication does not require a replication site to have a MaxSiteAge. Therefore, it is likely that one or more Release Replication sites will have a MaxSiteAge of 0 (zero), which is the default value recorded for a site if no

MaxSiteAge or DefaultSiteAge is specified. When changing from Release Replication to Scheduled Replication, the **-defaultsiteage** option *must* be used to set a DefaultSiteAge if any replication site does not have a MaxSiteAge and no DefaultSiteAge exists for the source fileset; otherwise, the **fts setrepinfo** command fails. If the command fails for this reason, reissue it, specifying a DefaultSiteAge with the **-defaultsiteage** parameter.

(See "Setting Replication Type and Parameters" on page 165 for details about the syntax of the **fts setrepinfo** command. See Table 8 on page 164 for information about which of the command's options apply to the two types of replication and the dependencies between the parameters.)

Note: If you switch from Scheduled Replication to Release Replication for a fileset, you need to verify that there is a replica on the same site as the read/write source.

Listing Replication Type and Parameters: You can use the **fts lsfldb** or **fts lsft** command to list information on the replication type and replication parameters defined for a fileset. (See "Chapter 7. Managing Filesets" on page 179 for more information on the options and output associated with these commands.)

```
$ fts lsfldb [-fileset  {name | ID}] [-server machine] \
[-aggregate name] [-locked ]
$ fts lsft [{-path  {filename | directory_name} | -fileset  { name | \
 ID}}] [-server machine]
```

Adding and Removing Replication Sites

After you define the replication parameters that are associated with a read/write fileset, you must define the sites (File Server machines and aggregates) where read-only replicas of the fileset are to be stored. Use the **fts addsite** command to add a replication site for a read/write fileset. If you define an incorrect site or decide at some later time that you no longer want a replica to be stored at a specific site, you can use the **fts rmsite** command to remove a replication site for a fileset.

With Release Replication, you must choose the File Server machine on which the read/write fileset resides as the first replication site. You can define one replication site on the same File Server machine and aggregate as the read/write fileset, and you can define up to 15 additional replication sites that are not on the same File Server machine and aggregate as the read/write fileset, for a maximum of 16 possible replication sites.

Note that you can store only a single read-only version of a given read/write fileset on any File Server machine. The **fts addsite** command prevents you from defining multiple replication sites for a fileset on the same File Server machine. Also, it is not possible to place a replica of a fileset on a File Server machine in a different cell.

Adding a Replication Site: To add a replication site for a fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Select a replication site for the read/write fileset that is to be replicated. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate; the **fts lsft** command can be used to check the size of the read/write fileset.

```
$ fts aggrinfo -server machine -aggregate name
```

3. Enter the **fts addsite** command to add the replication site. If Release Replication is being used, you must define the first replication site on the same File Server machine as the read/write fileset.

   ```
   $ fts addsite -fileset { name | ID} -server   machine \
   -aggregate name [-maxsiteage interval]
   ```

   The **-maxsiteage** *interval* option can be used to override the DefaultSiteAge set with the **fts setrepinfo** command.

Repeat the **fts addsite** command for each replication site you want to define for the fileset.

Removing a Replication Site: To remove a replication site for a fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

   Note: If the fileset uses Release Replication and you are using the **fts rmsite** command to remove the replication site and replica on the same machine as the read/write fileset, you must also be included in the **admin.ft** file on the machine specified by the **-server** option. Be aware that, if you remove the read-only replica on the same machine as the read/write fileset, all other read-only replicas of that fileset become unavailable after the expiration of the fileset's FailAge.

2. Enter the **fts rmsite** command to remove a replication site and to instruct the Replication Server at the site to remove the read-only replica of the fileset. If the fileset uses Release Replication and the replication site is on the same File Server machine as the read/write fileset, the **fts rmsite** command itself removes the replica. Note that you must specify the aggregate ID with the **-aggregate** option if the aggregate on which the replica resides is not currently exported or has been detached with the **dfsexport** command.

   ```
   $ fts rmsite -fileset { name | ID} -server \
   machine -aggregate   name
   ```

Repeat the **fts rmsite** command for each replication site and read-only fileset you want to remove for the fileset.

Creating Read-Only Filesets

The following subsections describe how to

- Use Release Replication to replicate a fileset. You issue the **fts release** command to release a new replica of a fileset on the same File Server machine as the read/write version of the fileset. The Replication Servers at the fileset's other replication sites then update the replicas at their respective sites to match the replica on the read/write fileset's machine.
- Use Scheduled Replication to replicate a fileset. Although Scheduled Replication is automatically performed according to the replication parameters you define, you can use the **fts update** command to request an immediate update of all replicas of a fileset or of only the replica at a given site. The Replication Servers at the specified replication sites immediately begin to update their respective

replicas to match the version of the read/write fileset that exists at the time the command issued. The Replication Servers ignore the MinRepDelay parameter associated with the fileset.

Using the **fts release** or **fts update** command does not guarantee immediate access to data in the new version of a replica. A Cache Manager continues to provide data cached from the old version of the replica until the MaxAge for the fileset expires or until the Cache Manager needs to access data from the replica that it has not already cached.

Moreover, the **fts release** and **fts update** commands direct the Replication Servers at a fileset's replication sites to begin updating the replicas stored at their sites. However, the Replication Servers update their replicas one at a time, in a serial rather than a parallel manner. Therefore, all of a fileset's replicas are not updated simultaneously.

Replicas are updated even if shared byte-range locks exist against data within the replica. Shared byte-range locks are granted by DFS to provide compatibility with applications that require them. However, such locks are not honored by DFS during the replica update process.

To attempt to gain immediate access to data in the new version of a replica, issue the **cm flush** or **cm flushfileset** command to flush the old data from the cache. These commands force the Cache Manager to discard and, as necessary, replace data it has cached from the replica. Until all replicas have been updated, you cannot directly force the Cache Manager to access data from the new version of the replica.

The operations described in the following subsections assume that the preparatory steps described in "Preparing for Replication" on page 162 have all been performed.

Using Release Replication to Create Read-Only Filesets

For a fileset that uses Release Replication, do the following to create or update the fileset's read-only replicas:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the read/write source fileset is stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the read/write source fileset is stored and for each machine on which a read-only replica is to reside. If necessary, issue the **bos lsadmin** command to verify the members of a list.
2. Use the **fts release** command to create or update the read-only replica of the read/write source fileset that resides on the same File Server machine as the source fileset. The Replication Servers at each replication site then update the read-only filesets at their respective sites to match this replica.

   $ fts release -fileset {*name* | *ID*}

   The **-fileset** *name* or *ID* option is the name or ID number of the read/write fileset whose replicas are to be updated.

   Use the **-wait** option to ensure that command will not complete (return a prompt) until all replicas are updated.

   $ fts release -fileset {*name* | *ID*} **-wait**

Using Scheduled Replication to Create Read-Only Filesets

For a fileset that uses Scheduled Replication, replication occurs automatically according to the parameters established previously with the **fts setrepinfo** command. You do *not* need to explicitly enter a command to initiate Scheduled Replication. However, you can use the **fts update** command to request an immediate update of a fileset's read-only replicas at any time. Issue the **fts update** command to request an immediate update of the replica at a specific replication site or of all replicas at all sites:

```
$ fts update -fileset {name | ID} {-all | -server machine}
```

The **-fileset** *name* or *ID* option is the name or ID number of the read/write fileset whose replicas are to be updated.

The **-all** option specifies that all replicas of the fileset indicated with the **-fileset** option are to be updated. Use the **-all** option or use the **-server** option.

The **-server** *machine* option names a specific File Server machine on which the replica of the fileset indicated with the **-fileset** option is to be updated. Specify the File Server machine's DCE pathname, its host name, or its IP address. Use the **-server** option or use the **-all** option.

Displaying Replication Status

The following subsections describe how to

- Display the status of the Replication Server (**repserver** process) on a File Server machine with the **fts statrepserver** command. Use this command to determine if the Replication Server at a replication site is functioning properly. If it is not, replicas stored on that machine may not be current.
- Display the statuses of all replicas of a fileset or the status of only the replica at a specific site with the **fts lsreplicas** command. Use this command to determine if the replicas of a fileset have been properly updated with the most recent version of the fileset.

Displaying the Status of a Replication Server

Enter the **fts statrepserver** command to determine if the Replication Server on a File Server machine is running:

```
$ fts statrepserver -server machine [-long ]
```

The **-long** option specifies that more detailed information about the Replication Server on the File Server machine indicated with the **-server** option is to be displayed. The additional information includes the status of each replica managed by the Replication Server.

Displaying the Statuses of Read-Only Filesets

Use the **fts lsreplicas** command to verify the statuses of a fileset's replicas at one or all of its replication sites:

```
$ fts lsreplicas -fileset {name |  ID} {-all  | -server machine}
```

The **-fileset** *name* or *ID* option is the name or ID number of the fileset whose replicas are to be checked.

The **-all** option specifies that all replicas of the fileset indicated with the **-fileset** option are to be examined. Use the **-all** option or use the -server option.

The **-server** *machine* option names a specific File Server machine on which the replica of the fileset indicated with the **-fileset** option is to be checked. Specify the File Server machine's DCE pathname, its host name, or its IP address. Use this option or use the **-all** option.

Creating Backup DCE LFS Filesets

A backup fileset is a single copy of the read/write version of a DCE LFS fileset. A backup fileset is stored at the same site as the read/write fileset on which it is based; it can serve as an online backup of the read/write fileset. Data in a backup fileset cannot be modified, but it can be read and copied. Backup versions of non-LFS filesets cannot be created.

Creating backup DCE LFS filesets has the following purposes:

- It is the first step in using the Backup System to copy a DCE LFS fileset permanently to tape.
- It allows you to create backup copies without interrupting a user's work.
- It enables users to restore deleted or changed data themselves. A backup version of a fileset captures the state of its read/write source at the time the backup is made. If you mount the backup version as a subdirectory of a user's home directory (with a suitable subdirectory name, such as **OldFiles** or **BackUp**), the user can restore files to the state they were in at the time of the backup without your help.

  If you create and mount backup filesets for your users, you need to explain to them how often you will make the backups and remind them that the data in their backup filesets cannot be changed. They can, however, copy the data to a directory in a read/write fileset and use it there.

An Overview of Backup Filesets

Although backup and read-only filesets are created in a similar manner, they serve a different purpose. Backing up a fileset does not make data available from multiple sites or reduce the load on a machine, as replication does. However, backing up a fileset is an efficient way to make previous versions of data such as user files available for restoration.

When you create a backup fileset, the backup fileset is filled with an array of pointers to the data housed in the read/write source. Then, the identities of the read/write source and the backup fileset are switched; the backup fileset becomes the read/write source, and the read/write source becomes the backup fileset. The sharing of data between the read/write and backup versions of a fileset results in significant disk space savings. (See "Data Sharing Among the Different Types of DCE LFS Filesets" on page 134 for more information on data sharing among the different types of DCE LFS filesets.)

A backup version preserves the exact state of its read/write source at the time the backup is created. For example, if you make a new version of each user's fileset every day at the same time, data that was deleted a week ago cannot be restored from the backup version because that backup version will have been overwritten six times since then. Similarly, if you make a new version of each user's fileset every day and a user revises a file three times during the same day, there is no way to

access the first and second revisions the next day; the backup version records only the third and final version of the file, which is the version that existed in the read/write fileset when the backup fileset was made.

A backup fileset must reside at the same site (File Server machine and aggregate) as its read/write source. It has the same name as the read/write version, with the addition of a **.backup** extension.

Backup Options

You can use the **fts clone** command to back up a single read/write DCE LFS fileset; you can use the **fts clonesys** command to back up multiple read/write DCE LFS filesets at one time. You can combine the options available with the **fts clonesys** command in different ways to indicate the filesets that are to be backed up; the following list summarizes the possible combinations. To back up

- All filesets in the local cell, specify no options
- All filesets in the local cell with a name beginning with the same character string (for example, **sys.** or **user.**), specify the string with the **-prefix** option
- All filesets on a File Server machine, specify the machine's name with the **-server** option
- Filesets on a specific aggregate on a File Server machine, specify both the **-server** and **-aggregate** options
- Filesets with a certain prefix on a specific File Server machine, specify both the **-prefix** and **-server** options
- Filesets with a certain prefix on a specific aggregate on a File Server machine, specify the **-prefix**, **-server**, and **-aggregate** options

You may want to make backup versions of particular filesets every day at the same time. You can issue the **fts clonesys** or **fts clone** commands at the console, or you can create a **cron** entry in the **BosConfig** file on each File Server machine where you want to back up filesets.

The following example creates a **cron** process called **backupusers** in the **BosConfig** file on the machine named **fs3**. The process executes every day at 5:30 a.m., making a backup version of every fileset in the cell's filespace that has a name that starts with **user**. The **-localauth** option allows the process, which runs unauthenticated, to use the DFS server principal of **fs3** to execute the privileged **fts clonesys** command.

```
$ bos create /.../abc.com/hosts/fs3 backupusers cron "dcelocal/bin/fts \
clonesys -prefix user-localauth" 5:30
```

Creating and Mounting Backup Filesets

To create and mount backup filesets, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on each machine on which a backup fileset is to reside, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of a fileset to be backed up resides. If necessary, issue the **bos lsadmin** command to verify the members on an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for each directory in which a backup fileset is to be mounted. If necessary, issue the **dcecp acl show** command to check the permissions for the directories.

3. Issue the **fts clone** command to create a single backup version of a read/write source fileset. The backup version is placed at the same site as the read/write source, with the same name as the source fileset but with the addition of a **.backup** extension.

   ```
   $ fts clone -fileset { name | ID}
   ```

   Issue the **fts clonesys** command to create a backup version of every read/write fileset that shares the same prefix or site. Each backup version is placed at the same site as its read/write source, with the same name, and with a **.backup** extension.

   ```
   $ fts clonesys [-prefix string] [-server machine] \
   [-aggregate name]
   ```

   The **-prefix** *string* option is the initial string in the name of each read/write fileset that you want to back up.

4. Enter the **fts crmount** command to create a mount point for each backup fileset. The contents of filesets are inaccessible until you perform this operation.

   ```
   $ fts crmount -dir directory_name -fileset
   name.backup
   ```

   The **-dir** *directory_name* option is the location for the root directory of the backup fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

   The **-fileset** *name***.backup** option is the full name of the backup fileset, with the **.backup** extension.

   Repeat the **fts crmount** command for each backup fileset created in the previous step. Note that you do not need to create a mount point for a backup fileset; for example, the DFS Backup System does not require a mount point to locate a backup fileset.

Using Mount Points

To make a DCE LFS or non-LFS fileset visible in the DCE namespace, you must use the **fts crmount** command to create a mount point for it in the DFS filespace. Mount points create the appearance of a single, seamless file system even though different filesets are stored on different File Server machines. A fileset's contents are visible and accessible as files and directories in the filespace only when the fileset is mounted; they are not available if the fileset is not mounted. Once its mount point is created, a fileset is automatically mounted; you do not need to take any further actions or explicitly mount it again at any time in the future.

A mount point is a specific type of symbolic link stored in the file system. It acts as an association between a directory location and a fileset. The root of a fileset contains an actual directory structure, to which the Cache Manager assigns the name of the fileset's mount point. A fileset's root directory provides a pathname for all of the files, subdirectories, and mount points contained in the fileset.

During file retrieval, the Cache Manager traverses a file's complete pathname until it finds the file. When it encounters a mount point in a pathname, the Cache Manager reads the mount point to learn which fileset is mounted at the mount point. It then contacts the Fileset Location Server to locate the fileset. As it traverses a

pathname, the Cache Manager interprets any additional mount points it finds until it reaches the fileset that contains the requested file.

Do not mount a fileset at more than one location in the file system. Creating multiple mount points can distort the hierarchical nature of the file system. The Cache Manager stores a single pointer to the directory that contains the root directory of each fileset; the Cache Manager can become confused about which pathname to follow when searching for a file in a fileset with multiple mount points. This is true even if you specify the full pathname of a file. Create multiple mount points for a fileset sparingly, in only a very limited number of troubleshooting and testing situations. Remove the extraneous mount points as soon as they are no longer necessary.

Different types of mount points exist. “Types of Mount Points” describes the different types of mount points in detail. Briefly, a mount point can be characterized by its

- Fileset type (read/write, read-only, or backup)
- Mount point type (regular, read/write, or global root)

The **fts lsmount** command can be used to examine mount points to determine their types and the filesets they name. The command displays the following message for each directory that is a mount point:

```
'directory_name' is a mount point for fileset 'fileset_name'
```

In the output, *directory_name* is the name of a directory (in this case, a mount point) that you specify, and *fileset_name* is the name of the fileset for which *directory_name* serves as a mount point. The command also provides the following information about the directory and fileset:

- A # (number sign) precedes the fileset name if the directory is a regular mount point.
- A % (percent sign) precedes the fileset name if the directory is a read/write mount point.
- An **!** (exclamation point) replaces the fileset name if the directory is a global root mount point.

Do not create a symbolic link that begins with a **#** (number sign) or a **%** (percent sign) character. Because a mount point is a special type of symbolic link that uses these characters internally to identify its type, the Cache Manager becomes confused if it encounters a normal symbolic link that begins with one of these characters.

The **fts delmount** command is used to remove mount points. The fileset that is referred to by a removed mount point remains in existence, but its contents are not accessible until another mount point is created for it.

Types of Mount Points

A mount point can be distinguished by its fileset type and its mount point type. The following subsections describe the characteristics of the different types of mount points.

Fileset Type

The first characteristic of a mount point determines which type of fileset is named in the mount point. A read-only or backup fileset has a **.readonly** or **.backup** extension; a read/write fileset has no extension.

When a mount point names a fileset with a **.readonly** or **.backup** extension, the Cache Manager uses only the specified version of the fileset. The Cache Manager never accesses the read/write version of a fileset when it encounters a mount point that names a read-only or backup version; if the explicitly named version is unavailable, the Cache Manager reports an error. However, depending on the mount point type, the Cache Manager can access the read-only version of a fileset when it encounters a mount point that names the read/write version (the fileset name with no extension).

If the Cache Manager resolves a pathname containing an explicit **.backup** fileset mount point and then encounters a regular fileset mount point underneath it, the Cache Manager automatically accesses the second fileset **.backup** if one exists instead of the readwrite. If a **.backup** fileset does not exist, the DFS client accesses the readwrite fileset.

Mount Point Type

The second characteristic determines the type of the mount point. The majority of mount points are *regular mount points*. When the Cache Manager encounters a regular mount point, it checks the version of the fileset that the mount point indicates—read/write, read-only, or backup. If the read-only or backup version is indicated, the Cache Manager attempts to access that version. If the read/write version is indicated, the Cache Manager evaluates the type of fileset in which the mount point itself resides:

- If the regular mount point for a read/write fileset resides in a read/write fileset, the Cache Manager attempts to access only the read/write version of the fileset. If the read/write version does not exist or is inaccessible, the Cache Manager cannot access the fileset.
- If the regular mount point for a read/write fileset resides in a read-only fileset, the Cache Manager first attempts to access a read-only version of the fileset. If the fileset is not replicated, the Cache Manager attempts to access the read/write version of the fileset. If the fileset is replicated but all of the replicas are unavailable, the Cache Manager cannot access the fileset; it does not attempt to access the read/write version of the fileset.

Regular mount points allow the Cache Manager to retrieve files better than other types of mount points because regular mount points allow the Cache Manager to access read-only filesets whenever possible. There are normally several different instances of the read-only version of a fileset, but there is only one instance of the read/write version. Because more read-only copies are usually available, it is better to access the copies as often as possible.

A much less common type of mount point is a *read/write mount point*. Read-write mount points must name the read/write version of a fileset. When the Cache Manager encounters a read/write mount point, it attempts to access only the read/write version of the fileset, regardless of the type of fileset in which the mount point resides. If the read/write version does not exist or is inaccessible, the Cache Manager cannot access the fileset.

You usually mount read/write filesets with regular mount points. A regular mount point is explicitly *not* a read-only mount point. The Cache Manager can still access the read/write version of a fileset when it encounters a regular mount point if no read-only versions of the fileset exist or if the Cache Manager is already on a read/write traversal path.

The least common type of mount point is the *global root mount point*. This type of mount point is used to mount the root of the DCE global namespace. (Because it is maintained for backward compatibility with other file systems, it is not normally used. This documentation provides no examples of its use.)

Manipulating Mount Points

The following subsections describe the commands used to create, list, and remove mount points.

Creating a Mount Point

To create a mount point, do the following:

1. Verify that you have the necessary permissions for the directory in which the fileset is to be mounted. If the directory resides in a DCE LFS fileset, you must have the **w** (write), **x** (execute), **c** control, and **i** (insert) ACL permissions for the directory; if necessary, issue the **dcecp acl show** command to list the permissions for the directory. If the directory resides in a non-LFS fileset, you must have the **w** and **x** permissions for the directory.
2. Enter the **fts crmount** command to create a mount point for a fileset:

   ```
   $ fts crmount -dir directory_name {-fileset  { name | ID} | \
    -global } [-rw ] [-fast ]
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

   The **-global** option indicates that the mount point is for the root of the DCE global namespace. Do not use this option; it exists for backward compatibility with other file systems.

   The **-rw** option specifies the type of the mount point as read/write. The Cache Manager accesses only the read/write version of the fileset. If the **-rw** option is used, the **-fileset** option must name the read/write version. Omit the **-rw** option to create a regular mount point.

   The **-fast** option specifies that the existence of the fileset indicated with the **-fileset** option is *not* to be verified. By default, **fts** verifies the existence of the fileset and displays a warning if it does not exist. The command always creates the mount point, regardless of whether the fileset exists.

   The following examples illustrate the creation of a mount point for a user fileset. Initially, only the user filesets named *user.pat* and *user.terry* are mounted in */.../abc.com/fs/usr*.

   ```
   $ cd /.../abc.com/fs/usr
   $ ls
   pat      terry
   ```

The **fts crmount** command is used to mount the fileset named *user.vijay* at */.../abc.com/fs/usr/vijay*. Because the -rw option is omitted, the fileset is mounted with a regular mount point.

```
$ fts crmount /.../abc.com/fs/usr/vijay user.vijay
$ ls
pat      terry    vijay
```

Listing a Mount Point

To list a mount point, do the following:

1. Verify that you have the **r** (read) permission for each mount point that you want to examine. If necessary, issue the **dcecp acl show** command to list the permissions for a mount point that resides in a DCE LFS fileset.
2. Enter the **fts lsmount** command to list information about one or more mount points:

   $ fts lsmount -dir *directory_name*...

   The **-dir** *directory_name* option specifies the name of each mount point about which you want to list information.

   The following example lists the mount point for the fileset *user.vijay*, which was created in the previous example. The # (number sign) in the output indicates that the mount point is a regular mount point.

   ```
   $ fts lsmount vijay
   'vijay' is a mount point for fileset '#user.vijay'
   ```

Removing a Mount Point

To remove a mount point, do the following:

1. Verify that you have the necessary permissions for each directory from which you want to remove a mount point. If a directory resides in a DCE LFS fileset, you must have the **w** (write), **x** (execute), and **d** (delete) ACL permissions for the directory; if necessary, issue the **dcecp acl show** command to list the permissions for the directory. If a directory resides in a non-LFS fileset, you must have the **w** and **x** permissions for the directory.
2. Enter the **fts delmount** command to remove one or more mount points:

   $ fts delmount -dir *directory_name*...

   The **-dir** *directory_name* option specifies the name of each mount point that you want to remove.

   The following example removes the mount point for the fileset *user.vijay*. The fileset itself is not deleted, just its mount point. As a result, the fileset is no longer visible or accessible in the DCE namespace.

   ```
   $ ls
   pat      terry    vijay
   $ fts delmount vijay
   $ ls
   pat      terry
   ```

Chapter 7. Managing Filesets

This chapter explains in detail how to manage filesets in DFS. It describes how to obtain information about filesets, aggregates, and partitions, how to set fileset quotas, and how to rename, move, dump, restore, and delete filesets. It also details how to verify and maintain the consistency of your filesets and how to recover from possible file system problems. Where applicable, the management of filesets in other file systems is discussed.

"Chapter 6. Making Filesets and Aggregates Available" on page 131 provides details about how to export aggregates and partitions, and how to create, replicate, backup, and mount filesets. It also provides a general overview of filesets and the differences between DCE LFS and non-LFS filesets. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for introductory information about filesets.)

Note: For the DCE 2.2 for AIX implementation, non-LFS filesets refer to AIX Journaled File Systems or AIX CD-ROM File Systems that have been exported to the DFS filespace and registered in the Fileset Location Database.

An Overview of Fileset Terminology

DCE LFS aggregates are similar to the disk partitions that are found in UNIX and other operating systems. However, a DCE LFS aggregate also supports specialized fileset-level operations, such as quota-checking and cloning, and low-level operations, such as logging of metadata.

Each DCE LFS aggregate exported to the DCE namespace can house multiple filesets; filesets stored on DCE LFS aggregates are referred to as DCE LFS filesets. Each exported non-LFS disk partition can house only a single fileset; file systems stored on non-LFS partitions are referred to as non-LFS filesets. Most of the specialized features supported for DCE LFS filesets are not supported for non-LFS filesets.

In DFS, only a single type of each non-LFS fileset is available: the version that was made available when the partition on which it resides was exported. However, three types of each DCE LFS fileset are available:

- A *read/write version* of a DCE LFS fileset contains modifiable versions of the files and directories in that fileset. Every DCE LFS fileset begins as a single read/write fileset. Other fileset types are derived from the read/write version by creating an exact copy, or replica, of all of the data in the read/write, source fileset. There can be only one read/write version of a fileset.
- A *read-only fileset* is a copy of a read/write, source DCE LFS fileset. Read-write filesets can be replicated and placed at various sites in the file system. Every read-only copy of a fileset shares the name of the source fileset, with the addition of a **.readonly** extension. If a read/write source changes, its read-only replicas can be updated to match. Every read-only copy of a read/write fileset is generally the same; however, replicas at different sites can differ in controlled ways according to the replication parameters associated with the fileset.
- A *backup fileset* is a clone of a read/write, source DCE LFS fileset. It is stored at the same site and with the same name as the source, with the addition of a **.backup** extension.

For both DCE LFS and non-LFS filesets, the Fileset Location Server (FL Server) maintains the Fileset Location Database (FLDB). The database records information about the location of all filesets in a cell. Every read/write fileset has an entry in the FLDB. Each entry for a DCE LFS fileset also includes information about the fileset's read-only and backup versions. For each fileset, the information in the entry includes fileset names, ID numbers, site definitions, and status flags for the sites.

Information about DCE LFS filesets is also stored in fileset headers at each site that contains a copy of the fileset. Fileset headers are part of the data structure that records the disk addresses on the aggregate of the files in the fileset. Fileset headers also record the fileset's name, ID number, size, status flags, and the ID numbers of its copies. Because the header records some of the same information that appears in the FLDB, the **fts** program can access the information in the header if the FLDB becomes unavailable. The FLDB entry and the fileset header must always be synchronized; any **fts** commands that affect fileset status automatically record the change in the appropriate FLDB entry.

(See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for detailed information about DCE LFS filesets, non-LFS filesets, and how the two types of filesets are created.)

Standard Options and Arguments

When using the **fts** commands to create, manipulate, or delete filesets, you can often add the **-verbose** option to receive detailed information from the **fts** program about its actions as it executes the command. This can be particularly helpful if an operation fails for a reason that you do not understand. The amount of additional information produced by the **-verbose** option varies for different commands.

The following options and arguments are common to many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the Reference part of this guide and reference for complete details about each command.)

- The **-fileset** *name* option is the complete name (for example, *user.sandy*) or ID number (for example, **0,,34692**) of the fileset to be used in the command.
- The **-server** *machine* option is the File Server machine to use with the command. Unless otherwise indicated, you can use any of the following to specify the File Server machine:
  - The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*)
  - The machine's host name (for example, **fs1.abc.com** or **fs1**)
  - The machine's IP address (for example, **11.22.33.44**)
- The **-aggregate** *name* option is the device name (for example, **/dev/lv01**), the aggregate name (for example, **lfs1** or **/usr**), or the aggregate ID (for example, **3** or **12**) of the aggregate or partition to be used in the command. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file.
- The **-cell** *cellname* option specifies the cell with respect to which the command is to be run (for example, *abc.com*). The default is the local cell of the issuer of the command.
- The **-noauth** option directs the **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody**

has the necessary privileges to perform any operation. (See “Chapter 4. Using Administrative Lists and Keytab Files” on page 95 for information about disabling DFS authorization checking.) If you use the **-noauth** option, do not use the **-localauth** option.

- The **-localauth** option directs the **fts** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server** (for example, */.../abc.com/hosts/fs1/dfs-server*). Do not confuse a machine’s DFS server principal with its unique **self** identity. (See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for information about DFS server principals.)

  Use the **-localauth** option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

The **-cell**, **-noauth**, and **-localauth** options are always optional.

Listing Fileset Information

The following commands are available for listing information about filesets:

- The **fts lsfldb** command is used to list information about filesets from the FLDB.
- The **fts lsheader** command is used to list information from fileset headers on File Server machines and aggregates.
- The **fts lsft** command is used to list information from both the FLDB entry and the fileset header of a specific fileset.

If you know only the name of a file or directory that is housed in a fileset, you can use the **fts lsquota**, **fts lsft**, and **cm whereis** commands to access information about the fileset. (See “Determining Other Fileset Information” on page 186 for complete instructions on using these commands with file or directory names.)

Listing FLDB Information

The **fts lsfldb** command is used to display information from entries in the FLDB for both DCE LFS and non-LFS filesets. By combining the command’s options in different ways, you can tailor the type of information to be displayed, as follows:

- To display every FLDB entry, do not supply any options.
- To display every FLDB entry that mentions a specific File Server machine as the site of any version of a fileset, specify the machine name with the -server option.
- To display every FLDB entry that mentions a specific aggregate on a specific File Server machine as the site of any version of a fileset, specify both the **-server** and **-aggregate** options.
- To display the FLDB entries for filesets with locked entries, use the -locked option alone or with the **-server** option (and optionally the **-aggregate** option).
- To display a single FLDB entry, specify the fileset name or ID number with the **-fileset** option.

To list information about fileset entries in the FLDB, enter the **fts lsfldb** command with the options described previously:

```
$ fts lsfldb [-fileset  {name | ID}] [-server machine] \
 [-aggregate name] [-locked ]
```

The **-locked** option lists information only for filesets with locked FLDB entries. Use the **-locked** option alone or with the **-server** option (and optionally the **-aggregate** option), or use the **-fileset** option to view the FLDB entry for a specific fileset.

Interpreting the Output

The output appears in the following order:

- The fileset's name.
- The fileset IDs of the read/write, read-only, and backup versions of the fileset.
- For each version of a fileset, a status flag of **valid** or **invalid** indicates whether it actually exists at some site. For the read-only version, it indicates whether a replication site is defined.
- The minimum and maximum advisory RPC authentication bounds for the fileset. There are two sets of bounds: one governing communications with Cache Managers in the local cell and another governing communications with Cache Managers in foreign cells.
- The number of sites at which a version of the fileset exists.
- An indicator if the FLDB entry is locked. The indicator is omitted if the entry is not locked.
- The replication parameters that are associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) where read/write (**RW**), read-only (**RO**), or backup (**BK**) versions of the fileset reside.
- For a read-only version, the MaxSiteAge defined for that site. For a read/write version, **0:00:00** is displayed because no MaxSiteAge is associated with that version.
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine or **<nil>** if no group owns the server entry.

If the output includes more than one FLDB entry, information about the filesets is displayed in alphabetical order by fileset name. The last line of the output displays the total number of entries that were successfully reported and the total number of entries that were not reported (the number of entries that failed).

Following is an example of the output that this command generates for a single DCE LFS fileset:

```
$ fts lsfldb user.terry
user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
Minimum local protection level: rpc_c_protect_level_none
Maximum local protection level: rpc_c_protect_level_pkt_privacy
Minimum remote protection level: rpc_c_protect_level_none
Maximum remote protection level: rpc_c_protect_level_pkt_privacy
number of sites: 1
 Sched repl: maxAge=2:00:00;failAge=1d0:00:00;reclaimWait=18:00:00;
 minRepDelay=0:05:00;defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Listing Fileset Header Information

The **fts lsheader** command displays the fileset header from every DCE LFS fileset on a File Server machine or one of its aggregates. You control the amount of information to be displayed by including the **-fast** option to display only the fileset ID number of each fileset, including the **-long** option to display all available information about each fileset, or omitting both options to display a single line of information about each fileset. Information about the filesets is displayed in numeric order by fileset ID number if the -fast option is used; otherwise, it is displayed in alphabetical order by fileset name. To view the header of a single DCE LFS fileset, use the **fts lsft** command, as described in “Listing FLDB and Fileset Header Information” on page 184 .

Non-LFS filesets do not have DCE LFS fileset headers. However, the **fts lsheader** command can still be used to display some local information, such as fileset ID numbers, that is stored in the *dcelocal***/var/dfs/dfstab** file on a File Server machine.

To list information from fileset headers, do the following:

- Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the filesets reside. If necessary, issue the **bos lsadmin** command to verify the members of the list.
- Issue the **fts lsheader** command to display information from fileset headers. Include the **-fast** or **-long** option to see less or more information.

  ```
  $ fts lsheader -server machine [-aggregate name] \
    [{-fast  | -long }]
  ```

Interpreting the Output

Following is an example of the output from this command when it is executed with the **-fast** option:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1 -fast
0,,196953
0,,196956
    .
    .
0,,199845
0,,199846
```

When you omit both the **-fast** and **-long** options, the command produces the following information:

- The File Server machine name, aggregate name, and aggregate ID number where the filesets reside.
- The total number of filesets on the aggregate.
- Each fileset’s name (with a **.readonly** or **.backup** extension, if appropriate).
- Each fileset’s ID number.
- Each fileset’s type (**RW** for read/write, **RO** for read-only, or **BK** for backup).
- Each fileset’s allocation usage and quota usage, in kilobytes.
- Each fileset’s status (**On-line**, **Off-line**, or an error indicator).
- The total number of filesets online, the total number of filesets offline, and the total number of filesets busy. A busy fileset is one upon which a fileset-related operation is currently in progress (for example, the fileset is being moved or cloned, or the Replication Server is currently forwarding changes from the fileset to read-only replicas).

Following is an example of the output from the command when it is run without the **-fast** or **-long** option:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1

Total filesets on server fs3 aggregate lfs1 (ID 10): 16
user.terry   0,,196953 RW   5071 K alloc  8421 K quota On-line
user.wvh     0,,196956 RW   4955 K alloc  9371 K quota On-line
    .
    .
Total filesets on-line 15; total off-line 1; total busy 0
```

When you include the **-long** option, the command displays the following additional information for each fileset:

- Whether it is a DCE LFS (**LFS**) or non-LFS fileset
- Information about the state of the fileset
- The ID numbers of the parent, clone, and backup filesets that are related to the fileset
- The ID numbers of the low-level backing and low-level forward filesets that are related to the fileset
- The version number of the fileset
- The allocation and allocation usage, in kilobytes, of the fileset
- The quota and quota usage, in kilobytes, of the fileset
- The day, date, and time when the fileset was created (replicated or cloned for a read-only or backup fileset)
- The day, date, and time when the contents of the fileset were last updated (this is the same as the creation time for a read-only or backup fileset)

Following is an example of the output from the command when it is executed with the **-long** option:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1 -long

Total filesets on server fs3 aggregate lfs1 (ID 10): 16
user.terry 0,,196953 RW LFS     states 0x10010005   On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    429496729 K alloc limit;      1252 K alloc usage
        15000 K quota limit;      9340 K quota usage
    Creation Tue Oct 15 16:45:16 1991
    Last Update Fri Nov 22 11:36:00 1991

user.wvh 0,,196956 RW LFS     states 0x10010005  On-line
    .
    .
Total filesets on-line 15; total off-line 1; total busy 0
```

Listing FLDB and Fileset Header Information

You can use the **fts lsft** command to display information from both the FLDB and the fileset header for a single fileset. The output from the **fts lsft** command consists of the output from the **fts lsheader** command with the **-long** option followed by the output from the **fts lsfldb** command. You can indicate the fileset about which information is to be displayed in one of two ways: by specifying the name of a file or directory that is stored in the fileset with the **-path** option, or by specifying the name or fileset ID number of the fileset with the **-fileset** option.

Because the **fts lsft** command retrieves information from the fileset header, you can examine the read-only or backup version of a DCE LFS fileset by adding the

.readonly or **.backup** extension to the name of the fileset specified with the **-fileset** option or by specifying the ID number of the read-only or backup version. An error message is displayed if the read/write version no longer exists and you fail to specify the **.readonly** or **.backup** extension with the name of the fileset.

If multiple read-only replicas of a DCE LFS fileset exist, you can use the **-server** option to indicate the name of the File Server machine that houses the specific replica to be examined. While all replicas of a fileset are identical by default, indicating a specific replica can be useful if network or hardware problems caused only some of a fileset's replicas to be updated. Omit the **-server** option to display information about the replica at the fileset's oldest read-only site. The **-server** option is always unnecessary when the command is used to examine the read/write or backup version of a fileset.

A read-only version of a DCE LFS fileset can exist independently of a read/write version if the **fts rmsite** command is not used to remove its site when the **fts delete** command is used to remove the read/write version. A backup version of a DCE LFS fileset can exist independently of a read/write version if the **fts delete** operation used to remove the read/write version is interrupted before completion; for example, if a system or hardware failure stops a delete operation after the read/write version is removed but before the backup version is removed. (See "Dumping and Restoring Filesets" on page 197 for more information about deleting DCE LFS (and non-LFS) filesets.)

Because non-LFS filesets do not have fileset headers, the **fts lsft** command displays much less fileset header information for non-LFS filesets than it does for DCE LFS filesets.

To display information about a fileset from both its FLDB entry and its fileset header, enter the **fts lsft** command:

```
$ fts lsft [{-path  {filename | directory_name} | \
-fileset  {name | ID}}] \
 [-server  machine]
```

The **-path** *filename* or *directory_name* option is the name of a file or directory in the fileset. Use the **-path** option or use the **-fileset** option to specify the name or ID number of the fileset; omit both options to display information about the fileset that houses the working directory.

The **-server** *machine* option names the File Server machine that houses the version of the fileset about which information is to be displayed. This option is useful for examining a particular read-only replica of a DCE LFS fileset for which multiple replicas exist.

Following is an example of the output for this command. The fileset ID number is used to indicate the fileset about which information is to be displayed; the leading 0 (zero) and commas are omitted from the ID number.

```
$ fts lsft -fileset 196953
```

```
user.terry 0,,196953 RW LFS     states 0x10010005 On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    429496729 K alloc limit;      1252 K alloc usage
        15000 K quota limit;      9340 K quota usage
    Creation Fri Oct 15 16:45:16 1993
    Last Update Mon Nov 22 11:36:00 1993
```

```
user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
Minimum local protection level: rpc_c_protect_level_none
Maximum local protection level: rpc_c_protect_level_pkt_privacy
Minimum remote protection level: rpc_c_protect_level_none
Maximum remote protection level: rpc_c_protect_level_pkt_privacy
number of sites: 2
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00;
  reclaimWait=18:00:00; minRepDelay=0:05:00;
defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Determining Other Fileset Information

The following subsections describe commands that are used to obtain information about a fileset when you know only its name, its ID number, or the name of a file or directory that it contains. The subsections include descriptions of the following commands:

- The **fts lsquota** command, which is used to determine the name of a fileset from the name of a file or directory that it houses. Note that the primary usage of the **fts lsquota** command is to list fileset quota information, as described in "Setting and Listing Fileset Quota" on page 191, not to determine fileset names.
- The **fts lsft** command, which is used to determine the ID number of a fileset from its name or from the name of a file or directory that it houses.
- The **fts lsfldb** command, which is used to learn the location of a fileset from its name or ID number.
- The **cm whereis** command, which is used to learn the location of a fileset from the name of a file or directory that it houses.

The **cm whereis** command is from the DFS **cm** command suite. (See "Chapter 8. Configuring the Cache Manager" on page 217 for more information about **cm** commands used to configure the Cache Manager; see the Reference part of this guide and reference for more detailed information about all **cm** commands.)

Learning Fileset Names from Files or Directories

To learn the names of filesets from the names of files or directories that they contain, enter the **fts lsquota** command with the **-path** option:

$ fts lsquota [-path {*filename* | *directory_name*}**...]**

The **-path** *filename* or *directory_name* option is the name of a file or directory in each fileset whose name you want to determine. You can include multiple files or directories from different filesets. Omit this option to learn the name of the fileset that houses the working directory.

The **fts lsquota** command produces a single line of output for each fileset that houses a named file or directory. Each line begins with the name of the fileset to which the information corresponds. (See "Setting and Listing Fileset Quota" on page 191 for a complete description of the command's output.)

The following example displays the name of the fileset that contains the directory named */.../abc.com/fs/usr/terry*:

```
$ fts lsquota /.../abc.com/fs/usr/terry
Fileset Name   Quota  Used  % Used  Aggregate
user.terry     15000  5071
34%    86% = 84538/98300
(LFS)
```

Learning Fileset ID Numbers from Fileset Names

To learn a fileset's ID number from its name, enter the **fts lsft** command with the **-fileset** option:

$ fts lsft -fileset { *name* | *ID*}

The **fts lsft** command displays numerous lines of output about the named fileset. The first line of output lists the fileset's name followed by its ID number. (See "Listing FLDB and Fileset Header Information" on page 184 for a description of the command's output.)

The following example displays the fileset ID number (**0,,196953**) of the fileset whose name is *user.terry*:

$ fts lsft -fileset user.terry

```
_____________________________________________________________
user.terry 0,,196953 RW LFS     states
0x10010005 On-line
    fs3.abc.com, aggregate
lfs1 (ID 10)
    .
    .
    .
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
_____________________________________________________________
```

Learning Fileset ID Numbers from Files or Directories

To learn a fileset's ID number from the name of a file or directory that it contains, enter the **fts lsft** command with the **-path** option:

$ fts lsft [-path {*filename* | *directory_name***}]**

The **-path** *filename* or *directory_name* option is the name of a file or directory in the fileset whose ID number you want to determine. Omit this option to learn the ID number of the fileset that houses the working directory.

The **fts lsft** command begins its output with the name and ID number of the fileset that houses the named file or directory. (See "Listing FLDB and Fileset Header Information" on page 184 for a description of the command's output.)

The following example displays the fileset ID number (again **0,,196953**) of the fileset that contains the current working directory:

$ fts lsft

```
_____________________________________________________________
user.terry 0,,196953 RW LFS     states 0x10010005 On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    .
    .
    .
```

```
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Learning Fileset Locations from Fileset Names or ID Numbers

To learn the location of a fileset from its name or ID number, enter the **fts lsfldb** command with the **-fileset** option:

$ fts lsfldb -fileset { *name* | *ID*}

The **fts lsfldb** command concludes its output with a list of the File Server machines that house existing versions of the fileset. (See “Listing FLDB Information” on page 181 for a thorough description of the command’s output.)

The following example displays the name of the File Server machine that houses the fileset named *user.terry*:

```
$ fts lsfldb user.terry
user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
number of sites: 1
 Sched repl:maxAge=2:00:00;failAge=1d0:00:00;reclaimWait=18:00:00;
 minRepDelay=0:05:00; defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Learning Fileset Locations from Files or Directories

To learn the locations of filesets from the names of files or directories that they contain, enter the **cm whereis** command:

$ cm whereis [-path {*filename* | *directory_name***}...]**

The **-path** *filename* or *directory_name* option is the name of a file or directory in the fileset whose location you want to determine. You can include multiple files or directories from different filesets. Omit this option to learn the location of the fileset that houses the working directory.

For each named file or directory, the **cm whereis** command displays the name of the cell in which the file or directory exists, the name of the fileset in which the file or directory resides, and the name of each File Server machine that houses a copy of the fileset. If the names of multiple machines are displayed, the file or directory resides in a read-only fileset, and replicas of the fileset are stored on each machine displayed by the command. Typically, only filesets that are in demand by numerous users (for example, filesets that house frequently accessed binary files) are replicated; filesets that are used by only a limited number of users (for example, users’ home filesets) are seldom replicated.

The following example displays location information about the fileset that houses the directory named */.../abc.com/fs/usr/terry*:

```
$ cm whereis /.../abc.com/fs/usr/terry
File '/.../abc.com/fs/usr/terry' resides in the cell
'abc.com', in fileset 'user.terry', on host fs3.abc.com.
```

Listing Aggregate and Partition Information

The following commands are available for listing information about any aggregate or partition (non-LFS aggregate) that is exported to the DCE namespace:

- The **fts lsaggr** command is used to list all exported aggregates or partitions on a File Server machine.
- The **fts aggrinfo** command is used to list information about the amount of disk space available on a specific aggregate or partition or on all exported aggregates and partitions on a File Server machine.

These commands are especially useful when exporting additional aggregates or partitions from a machine, when creating read/write or read-only filesets on an aggregate, or when moving filesets between machines.

Listing Aggregates and Partitions

The **fts lsaggr** command is used to list the following information about each exported aggregate or partition on a File Server machine. The information is specified for each aggregate and partition in the *dcelocal***/var/dfs/dfstab** file.

- The aggregate name, which is specified in the second field of the **dfstab** file
- The device name, which is specified in the first field of the **dfstab** file
- The aggregate ID, which is specified in the fourth field of the **dfstab** file
- The file system type, which is specified in the third field of the **dfstab** file

Note: You can issue the **dfsexport** command with no options on a File Server machine to list all aggregates and partitions currently exported from the local disk to the DCE namespace. Like the **fts lsaggr** command, no privileges are required to issue the **dfsexport** command with no options.

Enter the **fts lsaggr** command to list information about all of the exported aggregates and partitions on a File Server machine:

```
$ fts lsaggr -server machine
```

The following example shows that two non-LFS partitions and two DCE LFS aggregates are exported from the File Server machine named **fs1**:

```
$ fts lsaggr /.../abc.com/hosts/fs1
There are 4 aggregates on the server \
  /.../abc.com/hosts/fs1 (fs1.abc.com):
              /usr (/dev/lv02): id=3   (non-LFS)
              /tmp (/dev/lv03): id=4   (non-LFS)
              lfs1 (/dev/lfs1): id=10   (LFS)
              lfs2 (/dev/lfs2): id=11   (LFS)
```

Listing Disk Space on Aggregates and Partitions

The **fts aggrinfo** command lists information about the total amount of disk space and the amount of disk space that is currently available on exported aggregates and partitions. It can be used to obtain size information about a specific aggregate or partition or about all of the exported aggregates and partitions on a File Server machine. The command displays the following information about each aggregate and partition:

- The file system type (**LFS** for a DCE LFS aggregate, or **Non-LFS** for a non-LFS partition).

- The aggregate name.
- The device name.
- The number of kilobytes of disk space currently available on the aggregate or partition.
- The total number of kilobytes on the aggregate or partition (not including any reserved disk space).
- The number of kilobytes, if any, of reserved disk space on the aggregate or partition. DCE LFS reserves a variable amount of disk space on each aggregate for internal purposes (for example, to accommodate additional space required for fileset move and clone operations). Some non-LFS file systems also reserve some amount of overdraw disk space for administrative purposes.

The **df** command available in the UNIX operating system displays roughly the same information as the **fts aggrinfo** command for non-LFS partitions, exported DCE LFS aggregates, and locally mounted DCE LFS filesets.

Enter the **fts aggrinfo** command to display information about the disk space that is available on a specific aggregate or partition or on all aggregates and partitions on a File Server machine:

$ fts aggrinfo server *machine* **[-aggregate** *name*]

The following example displays information about the disk space that is available on the aggregates and partitions that are exported from **fs1**:

```
$ fts aggrinfo /.../abc.com/hosts/fs1
Non-LFS aggregate /usr (/dev/lv02): 24048 K free out of total 98304
     (10923 reserved)
Non-LFS aggregate /tmp (/dev/lv03): 11668 K free out of total 12288
     (1365 reserved)
LFS aggregate lfs1 (/dev/lfs1): 100537 K free out of total 101340
     (2048 reserved)
LFS aggregate lfs2 (/dev/lfs2): 79957 K free out of total 81920
     (2048
reserved)
```

Increasing the Size of a DCE LFS Aggregate

DCE LFS aggregates are created with the **newaggr** command. The initial size of an aggregate is specified with the command's **-aggrsize** option. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for more information about the **newaggr** command.)

The **growaggr** command is used to increase the size of an existing DCE LFS aggregate. On operating systems that support logical volumes, the **growaggr** command is useful for increasing the size of an aggregate when the size of the logical volume on which it resides is increased. It can also be used to increase the size of an aggregate that was deliberately created smaller than the partition or logical volume on which it resides.

The size of the aggregate can be made as large as the size of the partition on which it resides. To determine the total number of 1024-byte blocks on the partition on which an aggregate resides without changing the size of the aggregate, specify the **growaggr** command's **-noaction** option and omit its **-aggrsize** option. To increase the size of an aggregate to the total size of the partition on which it resides, omit both the **-aggrsize** and **-noaction** options.

The size of an existing aggregate cannot be decreased. A size specified with the command's **-aggrsize** option must be at least three DCE LFS blocks greater than the current size of the aggregate. (The number of bytes in a DCE LFS block is defined on a per-aggregate basis with the **-blocksize** option of the **newaggr** command when an aggregate is created.) Specify both the **-aggrsize** and **-noaction** options with the command to determine whether the size specified with the **-aggrsize** option is valid without changing the present size of the aggregate.

To increase the size of a DCE LFS aggregate, do the following:

1. Log in as **root** on the machine on which the aggregate that is to be enlarged resides.
2. Increase the size of the AIX logical volume.
3. Issue the **growaggr** command to increase the size of the aggregate. (See the Reference part of this guide and reference for more detailed information about the **growaggr** command.)

   ```
   # growaggr -aggregate name [-aggrsize blocks] [-noaction ]
   ```

   The **-aggregate** *name* option is the device name or aggregate name of the DCE LFS aggregate to be grown. These identifiers are specified in the first and second fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file.

   The **-aggrsize** *blocks* option is the total number of 1024-byte blocks to be available on the specified aggregate. The number of 1024-byte blocks specified with this option cannot exceed the total size of the disk partition on which the aggregate resides, and it must be at least three DCE LFS blocks larger than the current size of the aggregate. Omit both the **-aggrsize** option and the **-noaction** option to increase the size of the aggregate to the total size of the logical volume on which it resides. Specify both the -aggrsize option and the **-noaction** option to determine whether the size specified with this option is valid without changing the current size of the aggregate.

   The **-noaction** option directs the command to display the total number of 1024-byte blocks on the disk partition on which the specified aggregate resides, provided the **-aggrsize** option is not specified. If the -aggrsize option is specified with the **-noaction** option, the command determines whether the specified size is valid. The current size of the aggregate is not affected if the **-noaction** option is used.

Setting and Listing Fileset Quota

By default, every newly created DCE LFS fileset has a quota of 5000 kilobytes (5000 units of 1024 bytes each). The **fts setquota** command can be used to modify the quota of a DCE LFS fileset. The **fts lsquota** command can be used to examine the available quota and quota usage for a DCE LFS fileset.

Because it does not represent the amount of physical data stored on the fileset, the quota of a DCE LFS fileset can be larger than the size of the aggregate on which the fileset resides. Similarly and more commonly, the combined quota limits of all filesets on a DCE LFS aggregate can exceed the size of the aggregate. Assuming that all users who own filesets on an aggregate do not use all of their quota, you can allocate more quota than an aggregate actually contains to minimize user requests for additional quota. If additional quota is allocated to filesets that reside

on an aggregate whose size can be increased, the aggregate can be grown to accommodate the additional quota if necessary (see “Increasing the Size of a DCE LFS Aggregate” on page 190).

The size of a non-LFS fileset is always equivalent to the size of the partition on which it resides. In the UNIX operating system, you can use the **df** command to determine the size of a non-LFS partition. The **df** command can also be used to check the size of an exported DCE LFS aggregate, but it cannot be used to display the size of a DCE LFS fileset unless the fileset is mounted locally. In addition, although you can use the **fts lsquota** command to check the space that is used and available on a non-LFS fileset, you cannot use the **fts setquota** command to set the quota of a non-LFS fileset.

With both the **fts setquota** and **fts lsquota** commands, you can indicate the fileset whose quota you want to set or display either directly, by specifying the name (or ID number) of the fileset or, indirectly, by specifying the name of a file or directory located in the fileset. With the **fts lsquota** command, you can display quota information about multiple filesets by specifying either multiple fileset names (or ID numbers) *or* multiple file or directory names.

The **fts lsquota** command displays different information for DCE LFS and non-LFS filesets. For a DCE LFS fileset, the command displays the following information:

- The name of the fileset.
- The quota for the fileset (expressed as a number of kilobytes), the number of kilobytes currently in use on the fileset, and the percentage of the quota currently in use on the fileset.
- The percentage of disk space in use, the number of kilobytes in use, and the number of kilobytes available on the aggregate on which the fileset resides.
- An **LFS** indicator to show that the fileset is stored on a DCE LFS aggregate.

For a non-LFS fileset, the command displays the following information:

- The name of the fileset.
- Zeros for the quota, usage, and percentage used of the fileset. Ignore these values for a non-LFS fileset; refer instead to the corresponding values for the partition on which the fileset resides.
- The percentage of disk space in use, the number of kilobytes in use, and the number of kilobytes available on the partition on which the fileset resides.
- A **non-LFS** indicator to show that the fileset is stored on a non-LFS aggregate (partition).

Note again that the UNIX **df** command can be used to display the same information for the partition on which a non-LFS fileset resides.

Setting Quota for a DCE LFS Fileset

To set the quota for a DCE LFS fileset, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of the list.
2. Issue the **fts setquota** command:

```
$ fts setquota {-path  {filename | directory_name} \
|-fileset  { name |  ID}} -size kbytes
```

The **-path** *filename* or *directory_name* option is the name of a file or directory in the fileset whose quota you want to set. Use the **-path** option or use the **-fileset** option to specify the name or ID number of the fileset whose quota you want to set.

The **-size** *kbytes* option is the maximum amount of disk space that all of the files and directories in the read/write version of the fileset can occupy. This includes files and directories in the read/write version of the fileset that are actually pointers to disk blocks in the backup or read-only version of the fileset. Specify the value in kilobytes; a value of 1024 kilobytes equals 1 megabyte.

Listing Quota, Size, and Other Information for a Fileset

To display quota information about a fileset, enter the **fts lsquota** command:

```
$ fts lsquota [{-path  {filename | directory_name }... \
|-fileset  { name | ID}...}]
```

The **-path** *filename* or *directory_name* option is the name of a file or directory in each fileset whose quota you want to display. You can include multiple files or directories from different filesets; it is not necessary to name more than one file or directory from the same fileset. Use the **-path** option or use the **-fileset** option to specify the name or ID number of each fileset whose quota information you want to display. Omit both options to display information about the fileset that contains the working directory.

Following is an example of this command and its output. The fileset named *user.terry* has a quota of 15,000 kilobytes, 5071 kilobytes (34%) of which are currently in use. The fileset named *user.jean* has a quota of only 5000 kilobytes, almost all of which is currently in use. The **<<** and **<<WARNING** messages indicate that the fileset named *user.jean* is over 90% full; the same message is displayed for an aggregate that is over 97% full.

Ignore the values displayed in the fileset information columns for the non-LFS fileset, *user.jlw*. The quota and usage information for a non-LFS fileset is equal to the same information displayed for the partition on which the fileset resides. The partition that houses the fileset named *user.jlw* has 10,000 kilobytes available; 8448 kilobytes, or 84% of the partition, are currently in use.

```
$ fts lsquota /.../abc.com/fs/usr/terry /.../abc.com/fs/usr/jean /.../abc.com/fs/usr/jlw
Fileset Name   Quota  Used  % Used  Aggregate
user.terry     15000  5071     34%    86% = 84538/98300 (LFS)
user.jean       5000  4955     99%<<  92% = 87436/98300 (LFS)
   <<WARNING
user.jlw           0     0      0%    84% = 8448/10000 (non-LFS)
```

Setting Advisory RPC Authentication Bounds for Filesets

You can set upper and lower advisory RPC authentication bounds for any DCE LFS fileset. These bounds merve to bias a Cache Manager's initial RPC authentication level when transferring data to or from the fileset. The bound RPC authentication level values are stored in the FLDB by the **fts setprotectlevels** command. Currently these bounds are only advisory.

In operation, a Cache Manager contacts an FL Server to learn which File Servers house the required fileset (or replicas of the fileset). Along with the location, the Cache Manager also receives the upper and lower RPC authentication bounds for

that fileset. The Cache Manager then compares its initial RPC authentication level with the range defined by the advisory bounds. If the initial level falls within the range, the Cache Manager begins the process of negotiating an RPC authentication level with the File Server by using the initial level. If the initial level falls outside the range, the Cache Manager adjusts the initial level upward or downward to the closest bound value (though not below its own minimum setting) before beginning the process of negotiating an RPC authentication level.

For example, suppose the following values represent the Cache Manager and fileset authentication level settings:

- The Cache Manager initial RPC authentication level is set to packet.
- The fileset upper bound is set to packet privacy.
- The filset lower bound is set to packet integrity.

When the Cache Manager compares its initial level to the range defined by the fileset advisory bounds, it discovers that its initial level is set below the lower bound. The Cache Manager then adjusts its initial level to packet integrity and uses this RPC authentication level to begin the process of negotiating the RPC authentication level with the File Server. If the File Server upper bound is below the Cache Manager's initial level (adjusted through the fileset advisory bounds), the Cache Manager then lowers its initial level. Thus, the fileset bounds serve only to bias the selection of the RPC authentication level to a higher or lower level; however, the settings for the File Server and Cache Manager can override this bias.

Issue the **fts setprotectlevels** command to set advisory authentication bounds for filesets.

```
$ fts setprotectlevels –fileset  { name| ID}
[-maxlocalprotectlevel level]
[-minlocalprotectlevel   level]
[-maxremoteprotectlevel  level]
[-minremoteprotectlevel level]
[-cell cellname]
```

The following options set the various advisory RPC authentication bounds:

- The **-maxlocalprotectlevel** option specifies the upper bound for use by Cache Managers in the local cell.
- The **-minlocalprotectlevel** option specifies the lower bound for use by Cache Managers in the local cell.
- The **-maxremoteprotectlevel** option specifies the upper bound for use by Cache Managers in foreign cells.
- The **-minremoteprotectlevel** option specifies the lower bound for use by Cache Managers in foreign cells.

The *level* argument is set as follows:

- **0** or **rpc_protect_level_default** or **default**: Use the DCE default authentication level.
- **1** or **rpc_protect_level_none** or **none**: Perform no authentication.
- **2** or **rpc_protect_level_connect** or **connect**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **3** or **rpc_protect_level_call** or **call**: Authenticate only at the beginning of each RPC received.
- **4** or **rpc_protect_level_pkt** or **pkt**: Ensure that all data received is from the expected host.

- **5** or **rpc_protect_level_pkt_integrity** or **pkt_integrity**: Authenticate and verify that none of the data transferred has been modified.
- **6** or **rpc_protect_level_pkt_privacy** or **pkt_privacy**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

The following command sets the authentication values as follows:

- The maximum authentication level for communication with Cache Managers in the local cell is set to packet integrity.
- The minimum authentication level for communication with Cache Managers in the local cell is set to packet.
- The maximum authentication level for communication with Cache Managers in foreign cells is set to packet security.
- The minimum authentication level for communication with Cache Managers in foreign cells is set to packet security.

```
$ fts setprotectlevels -fileset richland.12 -maxlocalprotectlevel 5
-minlocalprotectlevel 4 -maxremoteprotectlevel 6
-minremoteprotectlevel 6
```

Renaming Filesets

You can use the **fts rename** command to change the name of the read/write version of any DCE LFS or non-LFS fileset. When you change the name of a fileset's read/write version, the names of the read-only and backup versions of the fileset are automatically changed accordingly. When you change the name of a fileset, you must also replace any mount points that reference versions of the fileset by the old name with mount points that indicate the new name.

To rename a fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the read/write fileset resides, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), **i** (insert), and **d** (delete) ACL permissions for the directory in which you need to replace the mount point. If necessary, issue the **dcecp acl show** command to list the permissions for the directory.
3. Issue the **fts rename** command to rename the fileset:

   ```
   $ fts rename -oldname  oldname -newname newname
   ```

   The **-oldname** *oldname* option is the current name of the read/write fileset.

   The **-newname** *newname* option is the new name of the fileset. The new name can be no longer than 102 characters. (See "Chapter 6. Making Filesets and Aggregates Available" on page 131 for more information on fileset naming conventions.)
4. Issue the **fts delmount** command to remove the mount point that indicates the fileset's old name:

   ```
   $ fts delmount -dir directory_name...
   ```

The **-dir** *directory_name* option is the name of each mount point you want to remove.

5. Issue the **fts crmount** command to create a new mount point that indicates the fileset's new name:

   ```
   $ fts crmount -dir directory_name -fileset { name | ID}
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the current working directory.

   The **-fileset** *name* or *ID* option is the name or ID number of the fileset to be mounted. Use the name you specified with the **fts rename** command in the earlier step.

Moving DCE LFS Filesets

The **fts move** command allows you to move read/write versions of DCE LFS filesets between aggregates on the same machine or between machines. Non-LFS filesets cannot be moved.

Filesets cannot be moved between sites in different cells. Furthermore, the physical disk on which a fileset resides cannot be moved from a machine in one cell to a machine in another cell with the expectation of simply running the **fts syncfldb** command to create an FLDB entry for the fileset in the new cell. There are two main reasons for these restrictions:

- The ACLs associated with the file and directory objects in a fileset are cell specific. There is no easy way to translate the ACLs associated with a fileset in one cell into an equivalent set of ACLs for another cell.
- Fileset IDs are unique to the local cell only. Any attempt to introduce a fileset from one cell into another cell risks a conflict between the newly introduced fileset and a fileset within the new cell that has the same fileset ID. A fileset ID conflict causes one of the two conflicting filesets to be inaccessible.

Note: You cannot dump and restore filesets between cells of the same name, even if you first remove the old cell and then recreate a new cell of the same name. For the purposes of fileset movement, two cells are different, regardless of whether they share a common name.

Read-write filesets are the only types of filesets that you can move. When you move the read/write version of a fileset, the backup version is automatically deleted from the read/write site; you cannot move the backup version of a fileset. Use the **fts clone** command to create a backup fileset at the new site. All read-only versions of a read/write fileset remain unaffected when the read/write source moves; use the **fts rmsite** and **fts addsite** commands to remove one replication site and add another. You do not need to change the mount point for a fileset when you move it.

Move filesets to another machine if their current machine or disk must be removed for repair. Consider moving filesets if an aggregate becomes full or if a File Server machine becomes overloaded.

Note: You cannot move a DCE LFS fileset that is also mounted locally (as a file system on its File Server machine) to a different File Server machine; you can move it only to a different aggregate on the same File Server machine.

To move a locally mounted DCE LFS fileset to a different server machine, remove its local mount point before attempting to move it. Also, because the backup version of a DCE LFS fileset is removed when the read/write version is moved, you cannot move a fileset, not even to another aggregate on the same File Server machine, if its backup version is mounted locally; you must remove the backup version's local mount point before moving the fileset.

To move the read/write version of a DCE LFS fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** files on both the source and destination machines, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entries for the source machine, the destination machine, and any machines on which replicas of the fileset reside. In addition, the source machine (specified with the **-fromserver** option) must be listed in the **admin.ft** file on the destination machine (specified with the -toserver option). If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

   If the fileset is to be moved to a new File Server machine, and a read-only version of the fileset using Release Replication already resides on the destination machine, enter the **fts rmsite** command to remove the replication site from the destination machine.

2. Enter the **fts move** command to move a read/write fileset from one site to another:

   ```
   $ fts move -fileset { name | ID} -fromserver source_machine \
   -fromaggregate  source_name -toserver dest_machine \
    -toaggregate dest_name
   ```

   The **-fromserver** *source_machine* option names the File Server machine on which the fileset currently resides. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

   The **-fromaggregate** *source_name* option is the aggregate on which the fileset is currently stored.

   The **-toserver** *dest_machine* option names the File Server machine to which the fileset is to move. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

   The **-toaggregate** *dest_name* option is the aggregate on which the fileset is to be stored after moving.

3. Enter the **fts lsfldb** command to confirm that the move was successful:

   ```
   $ fts lsfldb -fileset { name | ID}
   ```

4. Moving the read/write version of a fileset automatically deletes the backup version of the fileset if it exists at the read/write fileset's previous site. You can enter the **fts clone** command to create a new backup version at the new site:

   ```
   $ fts clone -fileset { name | ID}
   ```

Dumping and Restoring Filesets

The **fts dump** command converts the contents of a fileset to a byte stream format that can be stored in a file. Dumping a fileset does not affect its status in the FLDB or at the site from which it is dumped. You can dump any of the three types of DCE LFS filesets.

Note: AIX JFS and AIX CD-ROM File Systems may not be dumped or restored using the **fts** commands. The AIX DFS product only supports dump and restore of DFS LFS filesets.

Dumping is useful when you need to save a snapshot of a fileset (for example, when a fileset is removed but may later be restored). It is also useful if the read/write version of a fileset becomes corrupted; you can dump a backup or read-only version of the fileset and restore it as the read/write version, replacing the current, corrupted version.

You can perform a full or incremental dump of a fileset. A full dump of a fileset dumps the entire fileset as it currently exists. An incremental dump of a fileset dumps only those files from the fileset that have changed since a specified date and time; only those files with modification time stamps equal to or later than the specified time are dumped.

With DCE LFS filesets, you can also perform incremental dumps of only those files that have changed since a specified fileset version. Every DCE LFS fileset has a distinct version number that increments every time an operation is performed on the fileset or a file it contains (for example, when a file, directory, or ACL is modified). Each file in the fileset also has a version number. When an operation is performed on a file in a fileset, both that file and the fileset are marked with the current version number of the fileset plus one. When you do an incremental dump based on version, files in the fileset with version numbers equal to or greater than the version number you specify are dumped. (A DCE LFS fileset's version number is recorded in its fileset header; it has the same format as a fileset ID number. Use the **fts lsheader** or **fts lsft** command to view the current version number of a DCE LFS fileset.)

The **fts restore** command restores information from a previously dumped fileset back into the file system. Although you can dump any of the three types of DCE LFS filesets, you can restore a dump file only as a read/write fileset. When you restore a fileset, its creation date is set to the restoration date.

You can restore a dump file as a new DCE LFS fileset by specifying a name and site (File Server machine and aggregate) for the new fileset. The fileset is assigned an entry in the FLDB, and it receives the name that you specify with the command. The dump file must contain the full dump of a fileset if it is to be restored as a new fileset. After the fileset is restored, use the **fts crmount** command to create a mount point for the fileset, which makes it visible in the DCE namespace.

You can also restore a dump file as an existing read/write fileset (DCE LFS) by specifying the name and site of the existing fileset that is to be overwritten. The contents of the dump file overwrite the contents of the existing fileset. Include the **-overwrite** option with the **fts restore** command to specify that the existing fileset is to be overwritten; if you omit the **-overwrite** option, the command displays an error message and exits instead of overwriting the fileset.

If you are overwriting an existing fileset with an incremental dump, the fileset to be overwritten should initially have been restored as a new read/write fileset from a full dump. Also, both the dump file that is to be restored and the full dump that initially produced the read/write fileset that is to be overwritten must be dumps of the same fileset. Note that a full dump can be restored to overwrite an existing fileset, but the restored dump file overwrites all of the data in the existing fileset; an incremental dump cannot be restored to overwrite an existing fileset that was not created from the restoration of a full dump.

For the same reasons that you cannot move a fileset between sites in different cells, you cannot restore a fileset dumped in one cell to a site in another cell. (See “Renaming Filesets” on page 195 for information on the reasons for this restriction.)

Multiple incremental dumps of a fileset can be restored to overwrite the same existing fileset provided the following conditions are true:

- The fileset that is to be overwritten must not have been modified since its most recent restoration.
- The dump file that is to be restored must have been created *from* a date and time (as specified with the **-date** or **-version** option of the **fts dump** command) *no later* than the date and time at which the most recently restored dump of the fileset that is to be overwritten was dumped.
- The dump file that is to be restored must have been created *at* a date and time *later* than the date and time at which the most recently restored dump of the fileset that is to be overwritten was dumped.

The last two conditions specify that the span of time recorded in the incremental dump that is to be restored must overlap and extend the span of time recorded in the fileset that is to be overwritten. For example, suppose a full dump of a fileset was made on 1 February, an incremental dump from 31 January was made on 7 February, and a second incremental dump from 6 February was made on 14 February. The only possible way to restore all three dump files is to restore the full dump to a new read/write fileset, overwrite the new fileset with the incremental dump made on 7 February, and then overwrite the fileset with the incremental dump made on 14 February. Other sequences of restore operations involving all three dumps are very likely to result in some or all of the data in the fileset being inaccessible or inconsistent.

When restoring a dump file as a DCE LFS fileset, you can use the **-ftid** option to specify the fileset ID number that is to be associated with the fileset. If you are restoring to a new DCE LFS fileset, omit the **-ftid** option to let the FL Server allocate a new ID number; if you are overwriting an existing DCE LFS fileset, omit the option to retain the fileset’s current ID number. Specify a fileset ID number only if you are sure you can specify an unused ID.

Note: The contents of a fileset are unavailable during a dump operation. For this reason, you may want to dump only the backup version of a fileset, which does not interrupt access to the read/write and read-only versions.

Dumping a Fileset

To dump a fileset, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset is stored. If necessary, issue **bos lsadmin** to verify the members of the administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which you wish to store the dump file. If necessary, issue the **dcecp acl show** command to list the permissions for the directory.
3. Enter the **fts dump** command to dump the fileset:

```
$ fts dump -fileset { name | ID} {-time { date | 0} | \
 -version number} \
[-server machine] [-file filename]
```

The **-time** *date* or **0** option specifies a full or incremental dump. Use the **-time** option or use the **-version** option. There are three valid entries for the **-time** option:

- The **0** entry causes a full dump of the current version.
- The *mm/dd/yy* entry causes an incremental dump from 12:00 a.m. on the indicated date.
- The *mm/dd/yy hh:mm* entry causes an incremental dump from the specified time on the indicated date. The time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). Surround the entire argument with ″ ″ (double quotes) because it contains a space (for example, ″**11/22/97 20:30**″).

The **-version** *number* option specifies an incremental dump of the indicated version of the fileset. A fileset's version number is incremented with every change to the fileset or a file that it contains. Use the **-version** option or use the **-time** option. The **-version** option can be used *only* with DCE LFS filesets.

The **-server** *machine* option names the File Server machine that houses the version of the fileset to be dumped. This option is useful for dumping a particular read-only replica of a DCE LFS fileset for which multiple replicas exist.

The **-file** *filename* option specifies the complete pathname of the file in which the dump is to be stored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is sent to standard output (**stdout**).

Restoring a Dump File to a New Fileset

To restore a dump file to a new fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to be stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset is to be stored. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **r** (read) ACL permission for the dump file and the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the mount point for the new fileset is to be created. If necessary, issue the **dcecp acl show** command to list the permissions for the objects.
3. Select a site (an aggregate on a File Server machine) for the fileset. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate on which the fileset is to be placed:

   ```
   $ fts aggrinfo -server machine -aggregate name
   ```

4. Enter the **fts restore** command to restore the dump file to a new fileset:

   ```
   $ fts restore -ftname name -server machine -aggregate name \
    [-file   filename] [-ftid  ID]
   ```

   The **-ftname** *name* option specifies the name to be assigned to the restored fileset. The name can be no longer than 102 characters. (See “Chapter 6. Making Filesets and Aggregates Available” on page 131 for more information on fileset naming conventions.)

   The **-file** *filename* option specifies the complete pathname of the file that is to be restored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is taken from standard input (**stdin**).

The **-ftid** *ID* option specifies the fileset ID number that is to be assigned to the fileset. If this option is omitted, a new ID number is allocated for the fileset. Use this option with great care; make sure the fileset ID number that you specify is not in use.

5. Issue the **fts crmount** command to create a mount point in the file system for the new fileset, making its contents visible to other users:

   ```
   $ fts crmount -dir directory_name -fileset {name | ID}
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

Restoring a Dump File by Overwriting an Existing Fileset

To restore a dump file by overwriting an existing fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to be stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be overwritten is stored. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **r** (read) ACL permission for the dump file. If necessary, issue the **dcecp acl show** command to list the permissions for the file.
3. Enter the **fts restore** command to restore the dump file over an existing read/write fileset, using the **-overwrite** option to overwrite the current contents of the fileset:

   ```
   $ fts restore -ftname name -server machine -aggregate name \
   [-file filename] [-ftid ID] [-overwrite]
   ```

   The **-ftname** *name* option specifies the name of the fileset that is to be overwritten.

   The **-file** *filename* option specifies the complete pathname of the file that is to be restored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is read from standard input (**stdin**).

   The **-ftid** *ID* option specifies the fileset ID number that is to be assigned to the fileset. If this option is omitted, the current ID number of the existing fileset is retained. Use this option with great care; make sure the fileset ID number you specify is not in use. Use this option *only* when restoring a dump file as a DCE LFS fileset; omit this option when restoring a dump file as a non-LFS fileset.

   The **-overwrite** option specifies that the restored fileset can overwrite an existing fileset. If this option is omitted, the command refuses to overwrite an existing fileset. You must use this option to overwrite a previously restored version of a fileset with a dump file that contains an incremental dump of the same fileset or to restore a dump file as a non-LFS fileset.
4. If read-only copies of the former read/write fileset exist, use the **fts update** command to replace them with replicas of the new fileset. If a backup version exists, use the **fts clone** command to replace it with a backup version of the new fileset.

Removing DCE LFS Filesets

You can use the **fts delete** command to remove read/write and backup DCE LFS filesets. You can use the **fts rmsite** command to remove replication sites and instruct the Replication Servers at the sites to remove the read-only DCE LFS filesets. You can remove a read/write version without removing its read-only versions, and vice versa. You can also remove the backup version without removing the read/write version by including the **.backup** extension on the name of the fileset that is to be removed with the **fts delete** command. However, the backup version is automatically removed when the read/write version is removed. (Note that it is possible for a backup version to remain after its read/write source is deleted if a delete operation is interrupted prior to completion.)

If no other versions of any kind exist when you remove the last version of a DCE LFS fileset, the entire FLDB entry for the fileset is removed. When you remove the last version of a DCE LFS fileset, you also need to remove its mount point with the **fts delmount** command so users do not continue to try to access the fileset's contents. It is often better to remove a fileset's mount point before deleting the fileset; removing the mount point first ensures you that no users are accessing the fileset when it is deleted.

If you remove a read/write version of a DCE LFS fileset and read-only copies still exist, the FLDB status flags for the read/write version and the backup version change to **invalid**. The fileset ID is still recorded for each type, so you can restore the read/write version later. However, when you remove a read/write version of a fileset with the **fts delete** command, you should normally also remove its read-only copies by removing its replication sites with the **fts rmsite** command.

If you remove a read-only DCE LFS fileset while other read-only sites still exist, the site information for the removed copy is deleted from the FLDB entry. If no other read-only sites exist, the status flag for the read-only version is also changed to **invalid**; the fileset ID is still recorded, so you can recreate the read-only version later. If no other versions exist, the entire FLDB entry for the fileset is removed.

Removing the backup version of a DCE LFS fileset frees the space that it occupied on disk and changes the backup status flag in the FLDB to `invalid`. Its fileset ID is still recorded, since under normal circumstances the backup version cannot be the last existing version. When you remove the backup version, you may also want to remove its mount point from the file system.

The **fts delete** command can be used only when an FLDB entry and a fileset header exist for the fileset. Other commands can be used to remove individual FLDB entries and fileset headers. (See "Restoring a Dump File to a New Fileset" on page 200 for more information about these commands.)

Note that, when you delete a read/write or backup version of a DCE LFS fileset, that version of the fileset no longer exists on disk. Before removing the read/write or backup version of a fileset, use the DFS Backup System to preserve a permanent copy of it on tape.

Note: You cannot remove a DCE LFS fileset that is also mounted locally (as a file system on its File Server machine). You must remove the fileset's local mount point before attempting to delete the fileset. Also, because the backup version of a fileset is removed when the read/write version is removed, you

cannot remove the read/write version of a fileset if its backup version is mounted locally; you must remove the backup version's local mount point before deleting the read/write version.

Removing a DCE LFS Fileset and Its Mount Point

To remove a DCE LFS fileset and its mount point, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset resides, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **d** (delete) ACL permissions for the directory in which the fileset is mounted. If necessary, issue the **dcecp acl show** command to list the permissions for the directory.

   Note: On DCE 2.2 for AIX, steps 3 and 6 can be performed by using the **rmfilesys.dfs** command or the AIX SMIT utility.
3. Enter the **fts delmount** command to remove the mount point for the fileset:

   $ fts delmount -dir *directory_name*

   The **-dir** *directory_name* option is the name of the mount point for the fileset.

   Perhaps you previously mounted the backup version as a subdirectory of the read/write fileset's root directory. Removing the read/write version's mount point makes a backup version's mount point inaccessible. If you mounted the backup version at a separate directory, you must explicitly remove the backup version's mount point again using the **fts delmount** command.
4. Issue the **fts rmsite** command to remove the fileset's replication sites and to instruct the Replication Servers at the sites to remove the read-only versions of the fileset:

   $ fts rmsite -fileset { *name* | *ID*} **-server** *machine*\
   -aggregate *name*

   Repeat the **fts rmsite** command to remove each of the fileset's replication sites. If Release Replication was used for the fileset, the **fts rmsite** command must also be used to remove the replication site and read-only replica on the read/write fileset's File Server machine.
5. Issue the **fts delete** command to remove the read/write version of the fileset. The backup version of the fileset is removed automatically.

   $ fts delete -fileset { *name* | *ID*} **-server** \
   machine **-aggregate** *name*
6. After removing the last copy of the fileset, enter the **fts delmount** command to remove the mount point for the fileset. Disregard this step if any copies of the read-only version remain in the file system and you want them to be accessible.

   $ fts delmount -dir *directory_name*

   The **-dir** *directory_name* option is the name of the mount point for the fileset.

   If you previously mounted the backup version as a subdirectory of the read/write fileset's root directory, removing the read/write version's mount point makes the backup version's mount point inaccessible. If you mounted the backup version at a separate directory, you must explicitly remove the backup version's mount point, again using the **fts delmount** command.

Other Commands for Removing Filesets

Under normal circumstances, always use the **fts delete** or **fts rmsite** command to remove a fileset; these commands automatically record the deletion in the FLDB. Under special circumstances, however, you may need to use the following commands. Keep in mind that, if the FLDB and the filesets are consistent with each other, these commands make them inconsistent. Never use these commands unless absolutely necessary.

- Use the **fts delfldbentry** command to remove an FLDB entry that mentions a particular fileset. If versions of the fileset still exist at sites, they are not affected. This is useful if you are certain that a fileset removal was not recorded in the FLDB, and you do not want to use the **fts syncfldb** and **fts syncserv** commands to synchronize the entire FLDB. Use the **fts lsfldb** or **fts lsft** command to determine if a fileset removal was recorded in the FLDB.
- Use the **fts zap** command when it is urgent that a fileset be removed from its site, but the FLDB is inaccessible (for example, if the FL Server is unavailable). You can then remove the fileset's entry from the FLDB by entering the **fts delfldbentry** command or by entering the **fts syncfldb** and **fts syncserv** commands to synchronize the FLDB. The **fts zap** command, like the **fts delete** command, cannot be used to remove a DCE LFS fileset that is also mounted locally; you must remove the fileset's local mount point before attempting to delete the fileset.

The following subsections provide brief descriptions of the syntax and use of these commands.

Removing a Fileset's FLDB Entry Without Removing the Fileset

To remove a fileset's FLDB entry without removing the fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine that houses a version of any fileset whose FLDB entry is to be removed. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Enter the **fts delfldbentry** command to remove the fileset entry from the FLDB. Because this command lets you remove multiple FLDB entries simultaneously, be careful to remove only those FLDB entries you no longer need.

   ```
   $ fts delfldbentry {-fileset  {name | ID} | -prefix string} \
    [-server machine] [-aggregate  name]
   ```

3. The **-prefix** *string* option specifies a character string of any length. Every FLDB entry that lists a fileset whose name begins with *string* is removed. If the **-server** and **-aggregate** options are specified, only entries for filesets on the specified server machine and the specified aggregate are removed. Use the **-prefix** option, or use the **-fileset** option to specify the name or ID number of the fileset whose FLDB entry is to be removed.

Removing a DCE LFS Fileset Without Updating Its FLDB Entry

To remove a DCE LFS fileset without updating its FLDB entry, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset to be removed resides. If necessary, issue the **bos lsadmin** command to verify the members of the administrative list.

2. Enter the **fts zap** command to remove the fileset without recording the removal in the FLDB:

   $ fts zap -ftid *ID* **-server** *machine* **-aggregate** *name*

   The **-ftid** *ID* option specifies the fileset ID number of the fileset that is to be removed.

Removing Non-LFS Filesets

When you remove a non-LFS fileset, it becomes inaccessible in the DCE namespace. However, it is still available on the local disk of the machine on which it resides. (You can use the appropriate command in your local operating system to remove the partition that houses the non-LFS fileset from the disk.)

Before you delete a non-LFS fileset, you should remove the fileset's mount point using the **fts delmount** command. To remove a non-LFS fileset from the DCE namespace, use the **fts delfldbentry** command to remove the entry for the fileset from the FLDB. This prevents the FL Server from reporting the location of the fileset to a Cache Manager that requests data from the fileset. The **fts delfldbentry** command lets you remove multiple FLDB entries simultaneously; be careful to remove only those FLDB entries you no longer need.

Then issue the **dfsexport** command with the **-detach** option to detach the non-LFS partition on which the fileset resides from the namespace; when you detach a partition, it is no longer exported. These steps make the fileset unavailable in the DCE namespace. After you issue the **dfsexport** command, remove the partition's entry from the */dcelocal***/var/dfs/dfstab** file to prevent it from being exported the next time the machine is rebooted; note that this occurs only if the **dfsexport** command is included in the machine's initialization file (**/etc/rc** or its equivalent).

Any of these steps performed alone makes the fileset inaccessible. However, you should always perform all of the steps whenever you remove a non-LFS fileset to prevent future problems, such as a mount point that refers to a fileset that is no longer exported.

To remove a non-LFS fileset and its mount point, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **d** (delete) ACL permissions for the directory in which the fileset is mounted. If necessary, issue the **dcecp acl show** command to list the permissions for the directory.

   Note: On DCE 2.2 for AIX, steps 3 through 7 can be done in one step with the **rmfilesys.dfs** command or the AIX SMIT utility.
3. Enter the **fts delmount** command to remove the fileset's mount point:

   $ fts delmount -dir *directory_name*

   The **-dir** *directory_name* option is the name of the mount point you want to remove.

   $ fts delmount -dir *directory_name*

   The **-dir** *directory_name* option is the name of the mount point you want to remove.

4. Issue the **fts delfldbentry** command to remove the fileset entry from the FLDB:

   ```
   $ fts delfldbentry {-fileset  { name | ID} | -prefix   string} \
   [-server machine] [-aggregate name]
   ```

   The **-prefix** *string* option specifies a character string of any length. Every FLDB entry that lists a fileset whose name begins with *string* is removed. If the **-server** and **-aggregate** options are specified, only entries for filesets on the specified server machine and the specified aggregate are removed. Use the **-prefix** option, or use the **-fileset** option to specify the name or ID number of the fileset whose FLDB entry is to be removed.
5. Log in as **root** on the machine on which the fileset resides.
6. Enter the **dfsexport** command with the **-detach** option to detach the partition from the DCE namespace:

   ```
   # dfsexpor -aggregate name  -detach
   ```

   The **-aggregate** *name* option specifies the device name or exported aggregate name of the partition to be detached.

   The **-detach** option indicates that the specified partition is to be detached.
7. Use a text editor to remove the partition's entry from the **dfstab** file. An entry for a non-LFS partition in the **dfstab** file has the following format:

   ```
   /dev/lv02  /usr  ufs  1  0,,18756
   ```

Locking and Unlocking FLDB Entries

The FL Server locks the FLDB entry for a DCE LFS or non-LFS fileset before the Fileset Server executes any operations on it. A fileset with a locked FLDB entry is not affected by any other fileset manipulation operations such as moving or backing up a fileset. This immunity from other operations prevents inconsistencies or corruptions that can result from multiple simultaneous operations on a fileset. You can use the **fts lock** command to lock an FLDB entry and prevent an **fts** command from accessing it. You may want to lock an entry when you suspect it may be in error to prevent anyone from writing to it while you check the problem.

If an **fts** command operation fails prematurely, the FLDB entries can remain locked, preventing you from executing commands that can correct the problems. You can use the **fts lsft** and **fts lsfldb** commands to examine locked FLDB entries; you can use the **fts unlock** command to unlock a specific FLDB entry.

The **fts unlockfldb** command lets you unlock a set of entries based on the options you provide:

- When you provide no options, all locked entries are unlocked
- When you specify a File Server machine with the **-server** option, all locked entries with that machine in a site definition are unlocked
- When you specify an aggregate with the **-aggregate** option and a File Server machine with the **-server** option, all locked entries with that aggregate and that machine in a site definition are unlocked

Determining Whether an FLDB Entry is Locked

To determine whether an FLDB entry is locked, issue the **fts lsfldb** command:

```
$ fts lsfldb -fileset {name | ID}
```

If the entry is locked, the word **Locked** appears on a line of the output of the command. The **fts lsft** command also displays the same line in its output if an entry it is examining is locked.

Locking an FLDB Entry

To lock an FLDB entry, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be locked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts lock** command to lock the entry:

   $ fts lock -fileset { *name* | *ID*}

Unlocking a Single FLDB Entry

To unlock a single FLDB entry, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be unlocked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts unlock** command to unlock the entry:

   $ fts unlock -fileset { *name* | *ID*}

Unlocking Multiple FLDB Entries

To unlock multiple FLDB entries, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of any fileset to be unlocked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts unlockfldb** command to unlock all entries or only those entries on a specified server, on a specified aggregate, or on both:

   $ fts unlockfldb [-server *machine***] [-aggregate** *name*]

Synchronizing the FLDB and Fileset Headers

In DFS, transparent file access is possible because the FLDB constantly tracks fileset locations. When the Cache Manager needs a file, it contacts the FL Server, which consults the FLDB to find the current location of the file. Therefore, the FLDB must accurately reflect the state of filesets on all File Server machines. To keep the FLDB accurate, all **fts** commands that affect fileset status automatically record the change in the appropriate FLDB entry.

Whenever you issue a command that changes the status of a fileset, the **fts** program does the following:

- Directs the FL Server to lock the FLDB entry; the lock advises other operations not to attempt to manipulate any of the fileset's versions (read/write, read-only, or backup). This prevents simultaneous operations.

- Directs the FL Server to set an intention flag in the FLDB entry to indicate the type of operation to be performed.
- Directs the Fileset Server to perform the operation on the fileset. It may set an **Off-line** flag in the header, making the fileset inaccessible to other operations. When the operation is completed, the fileset is again marked **On-line**.
- Directs the FL Server to record the changes from the Fileset Server's operation in the FLDB. When the operation completes, the lock is released and the intention flag is removed. The fileset is again available for further operations.

Errors can occur if you are forced to stop an operation with an abort signal or if a File Server machine or server process goes down after you issue an **fts** command but before the requested operation is complete. It is likely in these situations that the FLDB is not synchronized with the headers of filesets on File Server machines. The following symptoms indicate that the FLDB and fileset headers are not synchronized:

- Error messages indicate that the operation terminated abnormally.
- A subsequent **fts** operation fails because the initial failure left an FLDB entry locked.
- A subsequent **fts** operation fails because the initial failure left a fileset marked **Off-line**.

The **fts syncfldb** and **fts syncserv** commands are used to synchronize the FLDB and fileset headers. The **fts syncfldb** command examines the fileset header of each online fileset on a specified File Server machine (or, optionally, a specified aggregate or partition on that File Server machine). It then checks the FLDB entry associated with the fileset header to verify that the FLDB entry is consistent with the fileset header. If an FLDB entry is not consistent with its fileset header, the **fts syncfldb** command changes the FLDB entry to make it consistent with the fileset header. If no FLDB entry exists for an online fileset, the command creates one; if an FLDB entry exists for a nonexistent fileset, the command deletes that entry.

The **fts syncfldb** command also performs three additional functions:

- If it finds a backup fileset whose read/write source no longer exists at the same site, it displays a warning message.
- If it finds a fileset ID number that is larger than the value of the counter that is used by the FL Server when allocating fileset ID numbers, it records this ID number as the new value of the counter. The next fileset to be created receives a fileset ID number that is one greater than this number.
- If necessary, it increments or decrements the number of fileset entries recorded as residing on a File Server machine in the FLDB entry for the server.

Note that the **fts syncfldb** command exits if it encounters a fileset that is busy. A busy fileset is one on which a fileset-related operation such as a move, clone, or release is currently in progress.

The **fts syncserv** command inspects the FLDB entry of each fileset on a specified File Server machine (or, optionally, a specified aggregate or partition on that File Server machine). The command then checks that each fileset header is consistent with its FLDB entry. If the command finds an inconsistency between the fileset name found in the fileset header and the name found in the FLDB entry, the fileset header is renamed to reflect the name in the FLDB entry. If the command encounters a fileset marked as **Off-line**, but the fileset's FLDB entry lists it as being **valid**, the command places the fileset online.

To ensure that the FLDB and all fileset headers in your cell are synchronized, run the **fts syncfldb** command once for each File Server machine in your cell. Then run the **fts syncserv** command once for each File Server machine in your cell.

Note: The **fts syncfldb** and **fts syncserv** commands synchronize only the FLDB entries for filesets on aggregates that are exported from the DFS File Server at the time that the command is run. If fileset entries are found in the FLDB but the aggregate is not exported at that time, no action is taken.

Note that, while the **fts syncfldb** and **fts syncserv** commands are useful in error recovery, they do not, in general, recover all of the information that is stored with a fileset's entry in the FLDB. More specifically, if the FLDB entry for a DCE LFS fileset is removed somehow and then recreated with the **fts syncfldb** command, replication information associated with the fileset is not restored. The **fts syncfldb** and **fts syncserv** commands cannot reproduce replication information once the entry for a DCE LFS fileset is removed from the FLDB. You must use the **fts setrepinfo** and **fts addsite** commands to reconstruct the replication information.

Synchronizing Non-LFS Filesets

The **fts syncfldb** and **fts syncserv** commands can be used to ensure the consistency of non-LFS filesets. However, because non-LFS filesets do not have fileset headers, the effectiveness of the commands is limited. You may need to take a more active role in returning consistency to non-LFS filesets and their FLDB entries.

For example, because non-LFS filesets do not have fileset headers, the **fts syncfldb** command cannot determine the name of a non-LFS fileset that has no FLDB entry. If the command determines that it needs to create an FLDB entry for a non-LFS fileset, it generates a unique name of the form **SYNCFLDB-ADDED-***number*. You then need to issue the **fts rename** command to rename the fileset to its original name.

Synchronizing Fileset Information

To synchronize the FLDB and fileset headers, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on each machine that houses a version of any fileset stored at a site specified with the **fts syncfldb** or **fts syncserv** command. You must also be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine that houses a version of any fileset stored at a site specified with either command. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts syncfldb** command to make FLDB entries consistent with filesets that are stored at the specified site. *Repeat this command for every File Server machine in your cell.*

   $ fts syncfldb -server *machine* **[-aggregate** *name*]

   The **-aggregate** *name* option names the aggregate on the server machine specified with the **-server** option for which fileset headers and FLDB entries are to be compared. Do not use this option under normal circumstances; omitting it allows synchronization of all of the filesets on the machine specified with the **-server** option. Use it only when just a single aggregate needs to be synchronized.

3. Issue the **fts syncserv** command to make filesets that are stored at the specified site consistent with FLDB entries. *Repeat this command for every File Server machine in your cell.*

   $ fts syncserv -server *machine* **[-aggregate** *name*]

   The **-aggregate** *name* option names the aggregate on the server machine specified with the **-server** option for which fileset headers and FLDB entries are to be compared. Do not use this option under normal circumstances; omitting it allows synchronization of all of the filesets on the machine specified with the **-server** option. Use it only when just a single aggregate needs to be synchronized.

Verifying and Maintaining File System Consistency

Many operating systems use the **fsck** program to ensure file system consistency after a system failure. The **fsck** program checks the consistency of a file system and reports its findings. Optionally, it repairs problems that it finds in the file system. The **fsck** program (or its equivalent) is still used to return consistency to many types of non-LFS partitions.

DFS employs a log mechanism and an additional system application, the DFS Salvager, to ensure the consistency of DCE LFS aggregates. A log is kept of all changes made to metadata on a DCE LFS aggregate as a result of operations such as file creation and deletion. The metadata records the structure and organization of the file system. Each DCE LFS aggregate has its own log, which physically resides on the aggregate, where it is completely transparent to users.

The DFS Salvager returns consistency to a file system when the system is restarted by replaying the log. Under normal circumstances, replaying the log returns the file system to a consistent state. However, if the Salvager detects problems in the basic structure of the aggregate, if the log mechanism is damaged, or if the physical storage medium of the aggregate is suspect, replaying the log cannot restore consistency. In these cases, a system administrator must invoke the Salvager a second time to examine and repair the structure of the aggregate.

Overview of the DFS Salvager

The DFS Salvager is used to replay the log on an aggregate and, if necessary, to find and repair file system inconsistencies that cannot be repaired by replaying the log. Along with the normal consistency guarantees provided by the log mechanism, the Salvager performs the same type of functions as the **fsck** program in other operating systems: It reads the metadata that describes the contents of a file system, analyzes the internal organization and structure of the file system, and detects and repairs inconsistencies.

The Salvager is invoked with the *dcelocal***/bin/salvage** command. The command can be used to direct the Salvager to do the following:

- Recover an aggregate following a system restart by replaying the log on the aggregate. The Salvager's replaying of the log is referred to as running recovery on the aggregate (or simply recovering the aggregate). This is the normal production use of the Salvager. Unless the contents of the log or the physical structure of the aggregate is damaged, replaying the log is an effective guarantee of a file system's integrity.

- Verify the structure of an aggregate to determine if it contains any inconsistencies. Recovering an aggregate and then verifying its structure represents a cautious application of the Salvager. Note that the Salvager can also be used to verify an aggregate before recovery is run, but the Salvager typically finds problems with an unrecovered aggregate that it would not find were recovery run first. In general, the findings of the Salvager for an unrecovered aggregate are of questionable worth.
- Salvage an aggregate by attempting to repair any inconsistencies it finds in the structure of the aggregate. Because recovery eliminates inconsistencies in an undamaged file system, an aggregate is typically recovered before it is salvaged. It is usually a good idea first to recover and then to salvage an aggregate if a machine panics or experiences a hardware failure.

As noted, running recovery to return consistency to a file system at restart time is the normal application of the Salvager. When it is installed, DFS automatically updates the local **/etc/start.dfs** configuration file to include the commands necessary to recover each DCE LFS aggregate listed in the **dfstab** file when the system is restarted. The system administrator can use the Salvager to verify or salvage an aggregate in addition to or instead of running recovery, as the situation warrants.

It is important to distinguish between file system consistency and user data consistency. The Salvager reads file system metadata; it does not try to verify the contents of the files in that file system. The Salvager can verify that each block in a file is correctly attached to that file, but it cannot verify the actual contents of the blocks. In cases where the metadata associated with a file is damaged, the owner of the file needs to verify that the file's contents are intact.

For example, if a disk controller accidentally writes on the disk surface, the Salvager tries to repair any inconsistencies in the structure of the file system. However, it has no mechanism to guarantee the contents of any file. In this case, the Salvager identifies any files whose metadata was damaged. After the file system is salvaged, users can verify the contents of these files; files whose contents were damaged can be restored from backups made before the file system problems occurred.

Not all aggregates can be salvaged. In the case of extensive damage to the structure of an aggregate or damage to the physical disk that houses an aggregate, the Salvager cannot repair inconsistencies.

Differences Between the DFS Salvager and fsck

While the DFS Salvager performs a role similar to that of the **fsck** program, several major differences distinguish the two programs:

- Recovery is usually sufficient to restore file system consistency at boot time. For this reason, the Salvager is typically used only to recover, not to verify or salvage, an aggregate when the system reboots. An administrator needs to use the Salvager to salvage an aggregate only if file system damage is suspected; for example, if the log cannot be replayed successfully, if a fileset cannot be mounted, or if a controller or disk failure affects the file system.
- The Salvager does not normally prompt the issuer for direction. It asks for confirmation to proceed only if it suspects that the aggregate on which it is run is not a DCE LFS aggregate or if it finds that the size of the aggregate that is recorded on disk exceeds the capacity of the partition on which the aggregate resides. It never asks the issuer for direction on how to repair the file system, in

which respect it is similar to the **fsck -p** command. Because of this, it can be run in the background, and several Salvager processes can be run simultaneously.

Note that the **-force** option of the **salvage** command can be used to direct the Salvager to proceed with all operations without requesting confirmation. However, if the Salvager is run on an invalid aggregate, using the **-force** option can produce unexpected changes.

- The Salvager displays information about files to be restored based on problems it discovers when it verifies or salvages an aggregate. A complete list of files (with pathnames, if possible) is printed when the operation completes. This output helps the system administrator complete the recovery of the files in the repaired file system.

Using the DFS Salvager

The **salvage** command is used to direct the Salvager to recover, verify, or salvage the structure of an aggregate. Combine the command's **-recoveronly**, **-verifyonly**, and **-salvageonly** options as follows to specify the operations the Salvager is to perform on the specified aggregate:

Specify the -recoveronly option
To run recovery on the aggregate without attempting to determine or repair any inconsistencies found on it. Use this option to quickly return consistency to an aggregate that does not need to be salvaged. This represents the normal production use of the Salvager.

Specify the -verifyonly option
To determine whether the structure of the aggregate contains any inconsistencies without running recovery or attempting to repair any inconsistencies found on the aggregate. Use this option to assess the extent of the damage to an aggregate. The Salvager makes no modifications to an aggregate during verification. Note again that it is normal for the Salvager to find errors when it verifies an aggregate that has not been recovered.

Specify the -recoveronly and -verifyonly options
To run recovery on the aggregate and then analyze its structure without attempting to repair any inconsistencies found on it. Use these options if you believe that replaying the log can return consistency to the aggregate, but you want to verify the consistency of the aggregate after recovery is run. This approach is more cautious than recovering the aggregate without verification.

Specify the -salvageonly option
To attempt to repair any inconsistencies found in the structure of the aggregate without first running recovery on it. Use this option if you believe that the log is damaged or that replaying the log will not return consistency to the aggregate and may in fact further damage it. Under normal circumstances, do not salvage an aggregate without first recovering it.

Omit the -recoveronly, -verifyonly, and -salvageonly options
To run recovery on the aggregate and then attempt to repair any inconsistencies found in the structure of the aggregate. Omit these three options if you believe the log should be replayed before attempts are made to repair any inconsistencies found on the aggregate.

You cannot use the Salvager to recover or salvage an aggregate that is currently exported to the DCE namespace. If asked to perform either of these operations on an exported aggregate, the Salvager exits without performing the operation. If

necessary, use the **dfsexport** command to detach an aggregate from the global namespace before recovering or salvaging it. Note that the Salvager also exits if it is run on an aggregate that houses a locally mounted fileset.

The Salvager prompts for direction only if one of the following is true:

- The aggregate on which it is run contains a non-LFS superblock whose creation time is more recent than that of its DCE LFS superblock.
- The size of the aggregate that is recorded in its DCE LFS superblock exceeds the capacity of the partition on which the aggregate resides.

At the prompt, you can choose to cancel or continue the operation. If you continue the operation in either of these situations and the aggregate proves to be invalid, the operation can produce unpredictable results. The best response in either case is to cancel the operation and attempt to determine the cause of the problem. Note that the command's **-force** option can be used to direct the Salvager to proceed rather than prompt for confirmation in these cases.

Internal structures maintained by the Salvager require a minimum of 1 megabyte of swap space. However, the total amount of swap space required by the Salvager depends largely on the size of the aggregate being salvaged and the extent of the damage to the aggregate.

Recovering, Verifying, or Salvaging a File System

To recover, verify, or salvage a file system, do the following:

1. If the specified aggregate is to be recovered or salvaged, log in as **root** on the local machine or verify that you have both the **r** (read) and **w** (write) permissions for the aggregate. If the specified aggregate is to be verified, log in as **root** on the local machine or verify that you have the **r** (read) permission for the aggregate.
2. Ensure that the DCE LFS aggregate to be specified with the command is not exported and contains no locally mounted filesets.
3. Enter the **salvage** command to run recovery on the aggregate, verify the consistency of the aggregate, or attempt to repair the consistency of the aggregate:

   ```
   # salvage -aggregate name [-recoveronly ] \
   [{-verifyonly -salvageonly }] [-force ] [-verbose]
   ```

   The **-aggregate** *name* option is the device name or aggregate name of the DCE LFS aggregate that is to be recovered, verified, or salvaged. Note that the aggregate ID of the aggregate is not an acceptable value.

   The **-recoveronly** option directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate.

   The **-verifyonly** option directs the Salvager to verify the specified aggregate. The Salvager examines the structure of the aggregate to determine if it contains any inconsistencies, reporting any that it finds.

   The **-salvageonly** option directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate.

The **-force** option executes the Salvager in noninteractive mode. By default, the Salvager prompts for confirmation before proceeding in certain situations. Use this option to direct the Salvager to proceed with all operations without asking whether it should continue.

The **-verbose** option directs the Salvager to produce detailed information about the aggregate as it executes. The information is useful primarily for debugging purposes. Use this option alone or with any other combination of options.

Interpreting Salvager Output

The Salvager displays output on standard output and standard error. When the **salvage** command is first executed, the Salvager displays the device name of the aggregate on which it is to run and the operation it is to perform. For example, the Salvager displays the following message if it is instructed to recover an aggregate:

```
Will run recovery on device
```

If you use the Salvager to recover an aggregate and the log on the aggregate does not need to be replayed, the Salvager displays only the brief introductory information. If the log does need to be replayed and the Salvager can successfully recover the aggregate, the Salvager displays the following messages:

```
Recovery statistics
    statistics
Ran recovery on device
```

In the output, *statistics* consists of a few lines of information about the log and its replaying, and *device* is again the device name of the recovered aggregate. If recovery fails for any reason, the Salvager returns an appropriate exit code. (All Salvager exit codes are described at the end of this section.)

The Salvager can display much more output if it is used to verify or salvage an aggregate on which it finds metadata errors. As it verifies or salvages a damaged aggregate, it displays a message similar to the following for each fileset in which it encounters metadata problems:

```
In volume fileset (avl # integer)
   in anode (# integer)
       description
```

It displays the first line once for each fileset, repeating the second and third lines once for each problem anode in the fileset. An anode is an area on disk that provides information used to locate data such as files, directories, ACLs, and other types of file system objects. Each fileset contains an arbitrary number of anodes, all of which must reside on the same aggregate.

In the output, *fileset* is the name and ID number of each affected fileset; **avl #** *integer* indicates the anode for the fileset; **in anode (#** *integer***)** indicates the anode for a file or other object in the fileset; and *description* provides a brief description of the problem the Salvager found with the anode (and any actions it took as a result, if it is salvaging the aggregate).

When it has finished executing, the Salvager lists files whose metadata it found to be damaged, many of which it likely repaired if it salvaged the aggregate. For each file, it displays a line of the form

condition *fileset*: *pathname* volume index: *integer* anode index: *integer*

In the output, *condition* is a string that describes the state of the file or its metadata; *fileset* is the name of the fileset in which the affected file resides; and *pathname* is the pathname of the file, relative to the root directory of the fileset. Note that the Salvager may not be able to determine the fileset name or reconstruct the pathname for every file.

The **volume index:** *integer* and **anode index:** *integer* provide pointers to the anodes that are associated with filesets and files whose metadata was damaged (and possibly repaired). The **volume index** indicates the anode for the fileset; the **anode index** indicates the anode for the file. Anode-related information is not useful for verifying or restoring data on an aggregate, but it does serve to identify earlier messages displayed by the Salvager that are related to this file.

The following conditions accompany the files most in need of attention:

oughtRestore
: Files in which one or more block references in the associated anode were removed or changed. Because it is unlikely that such files contain all of their original data, these files should be restored from existing backups. This condition applies only to files on salvaged aggregates.

mayRestore
: Files to which modifications were made (for example, files whose ACLs or property lists were changed). The owners of these files should verify their contents, or a system administrator should simply restore them from backups if a directory listing indicates that they have not been modified since the last backup was made. This condition also applies only to files on salvaged aggregates.

zeroLinkCnt
: Files whose link counts should be 0 (zero). These files were deleted but not closed when the system crashed or were orphaned by the Salvager as it made repairs to the file system. The system will delete them when the aggregate is exported.

badLinkCnts
: Files whose link counts were inconsistent with the number of references found to them. These files should be examined, if possible, or simply restored.

The Salvager can list a file more than once if it finds that multiple conditions apply to the file. It can also display one or more additional conditions, but files with which the additional conditions are associated are usually already covered by one or more of the conditions just described.

The Salvager also returns one of various exit codes summarizing its actions and findings. It returns the exit codes in the form of bits, which it uses to indicate the state of the aggregate. It can set multiple bits but, in general, the higher the bit, the greater the severity of the aggregate's problems; the higher bit always takes precedence when interpreting the output. The Salvager can return the following exit codes:

All bits off
: The Salvager found no problems. It displays a message that includes **Done** and **Checks out**. The command need not be run again.

First bit (0x1) set
The Salvager found one or more problems. It displays a message that includes **Done** and **Some inconsistencies found**. Run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Second bit (0x2) set
The Salvager found one or more problems and fixed them. It displays a message that includes **Done** and **Some inconsistencies repaired**. The command need not be run again. (Note that, if the second bit is set, the first bit is typically set as well; because the higher bit takes precedence, you do not need to run the command again.)

Third bit (0x4) set
The Salvager found one or more problems and fixed some of them. It displays a message that includes **Incomplete** and **Some repairs made**. Some of the problems were more severe and require a subsequent salvage to be repaired; run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Fourth bit (0x8) set
The Salvager found the aggregate to be irreparably damaged. It displays a message that begins **Problem**. Use the **newaggr** command to reinitialize the aggregate, and reconstruct the data from existing backups if possible.

Fifth bit (0x10) set
The Salvager found some serious problem that prevents it from running; for example, the attempted recovery of the aggregate failed because of damage to the log, or the attempted salvage of the aggregate failed because the aggregate is not a DCE LFS aggregate or it is currently exported. The Salvager displays a message that begins **Problem**. Attempt to determine the cause of the problem.

Including the **-verbose** option with the **salvage** command causes the Salvager to produce more detailed information about the aggregate. Much of the additional information is useful primarily for debugging purposes.

Chapter 8. Configuring the Cache Manager

This chapter describes the configuration of any machine that is to serve as a DFS client machine. All DFS client machines must do the following:

- Run the set of modifications known as the *Cache Manager* in the kernel. The Cache Manager enables the client to access DFS. You can control several aspects of Cache Manager and client performance by configuring the Cache Manager as described in this chapter.
- Run the **dfsd** process, which initializes the Cache Manager by transferring DFS configuration information into kernel memory.
- Run the **dfsbind** process, which resolves CDS pathnames for the Cache Manager and accesses user authentication information necessary for effective communications with server machines.
- Provide local disk space or memory sufficient to house configuration information and a cache.

Note: On DCE 2.2 for AIX, use the **config.dfs** command or the AIX SMIT utility to configure the DFS client. See the *IBM DCE for AIX, Version 2.2: Quick Beginnings* for more information.

An Overview of the Cache Manager

The Cache Manager fetches and caches files from File Server machines on behalf of application programs that are running on client machines. To locate a file to be retrieved, it first contacts the Fileset Location Server (FL Server) to learn the location of the fileset that houses the file; to retrieve the file, it then contacts the File Server machine that houses the file. The File Exporter on the File Server machine delivers the file, which the Cache Manager stores in the client machine's cache, which is an area of local disk or memory designated for temporary storage. Once the data is cached locally, the Cache Manager can access it as quickly as it can a local file.

The Cache Manager verifies that its cached files match the central copies at File Server machines by keeping the tokens that the File Exporters send with the files. A token acts as the File Exporter's promise to contact the Cache Manager if the centrally stored copy of a file changes while the Cache Manager has a cached copy. If the central copy changes, the File Exporter revokes the token; the Cache Manager sees that the token is revoked and retrieves the new version of the file when an application program next requests data from it.

Cache Manager Processes

The **dfsd** process controls the Cache Manager, which runs in the kernel. You can add options to the *dcelocal***/bin/dfsd** command and modify the *dcelocal***/etc/CacheInfo** file on the client machine to customize DFS cache parameters.

You can control several Cache Manager features with **dfsd** options. The **dfsd** process must be invoked at or after system reboot on all DFS client machines; it is recommended that the **dfsd** command be added to the proper initialization file (**/etc/rc** or its equivalent). The **dfsbind** process must be run before the **dfsd** process; it is recommended that the *dcelocal***/bin/dfsbind** command also be added to the proper initialization file.

Note: With DCE 2.2 for AIX, the **start.dfs** file is updated to include the starting of **dfsd** and **dfsbind** when a DFS client is configured. Refer to *IBM DCE for AIX, Version 2.2: Quick Beginnings* for information about configuring DFS and starting up of DCE at system startup on AIX.

Cache Manager Files

The **CacheInfo** file, which is manually created during DFS client installation, is composed of three fields separated by colons: the first field specifies where the DCE global namespace is mounted; the second field names the cache directory where the Cache Manager creates its cache files; and the third field specifies the cache size in 1024-byte (1-kilobyte) blocks.

The Cache Manager creates and maintains the following files, which are not intended for direct use. You can cause the kernel to panic if you accidentally modify any of these files; if this happens, rebooting the machine should restore normal performance. Note that these files exist only on client machines that use disk caching; a machine that uses a memory cache maintains in memory the cache information the files contain.

CAUTION:
Never **directly modify or delete these files; this can cause the kernel to panic.**
Always **use the commands provided with DFS to alter these files.**

- Multiple *dceloca*l**/var/adm/dfs/cache/V** *n* files, where *n* is a unique number for each file. By default, each **V** *n* file holds up to 64 kilobytes of a cached file; files larger than 64 kilobytes are divided into multiple files. The number of **V** *n* files, or **V** files, depends on the cache size.
- The *dceloca*l**/var/adm/dfs/cache/CacheItems** file. The Cache Manager uses this binary file to record information about each **V** file, including its file ID number and data version number.
- The *dceloca*l**/var/adm/dfs/cache/FilesetItems** file. This binary file stores the fileset-to-mount point mapping for each fileset accessed. This mapping enables the Cache Manager to respond correctly to commands such as **pwd**.

Cache Manager Features You Can Customize

You can alter the following aspects of the Cache Manager configuration to achieve different levels of performance on different client machines:

- Disk or Memory Cache: You can direct the Cache Manager to use machine memory instead of disk space for caching.
- Chunk Size and Number: You can use several options with **dfsd** to alter the default size and number of chunks that compose a cache. With a disk cache, each chunk is called a **V** file; with a memory cache, each chunk is represented by a block of memory. The size and number of chunks can be modified to take advantage of fast networks or to compensate for slow networks.
- Cache Location: The standard location of the cache (which is at *dceloca*l**/var/adm/dfs/cache**) can be changed to take advantage of greater space availability on other partitions.
- Cache Size: The cache size influences how often the Cache Manager contacts File Server machines across the network. Increasing the cache size results in better performance because fewer cross-network calls are necessary.
- File Server and FL Server Preferences: The Cache Manager maintains entries that contain the machine specifications (either host names or IP addresses) and

preference ranks for File Servers and FL Servers. A File Server entry's rank determines the Cache Manager's preference for electing to access replicas at that address over replicas that are available through other network connections. Similarly, an FL Server entry's rank determines the Cache Manager's preference for querying the FLDB through a particular address. You can specify preferences for both types of server machines to bias the Cache Manager's selection process. A File Server or FL Server will normally have up to four entries in a Cache Manager's preference list, with each entry having a separate machine specification.

- The **setuid** Status: By default, the Cache Manager does not allow **setuid** programs from filesets to execute with **setuid** permission. You can enable **setuid** programs from specific filesets to execute with **setuid** permission; **setgid** programs on a fileset are enabled and disabled along with **setuid** programs.
- Device File Status: By default, the Cache Manager does not honor device files stored in filesets. You can instruct the Cache Manager to recognize device files from specific filesets.
- Cached File Versions: The DFS token mechanism guarantees that the Cache Manager uses the most current versions of files and directories. You can also force the Cache Manager to discard the versions you are using and fetch new versions from the File Server machine.
- Unstored Data: If the Cache Manager cannot contact a File Server machine to write data to it, it keeps the unstored data in the cache. It then continues to attempt to contact the File Server machine until it can store the data. You can list all of the data the Cache Manager cannot store, and you can force the Cache Manager to discard the data rather than to continue to try to contact unavailable File Server machines.
- RPC Authentication Levels: The Cache Manager and File Server default authentication levels are such that communications default to the packet integrity RPC authentication level. You can set two sets of initial RPC authentication levels and minimum RPC authentication levels; one set governs communications with File Servers in the local cell, while the second set governs communications with File Servers in foreign cells.
- Client Persistent Requests: By default, the Cache Manager does not run with the persistent requests parameter enabled. When this parameter is enabled, instead of immediately returning errors to clients, the Cache Manager retries the request for errors resulting in the loss of access to DFS file exporters. The Cache Manager retries the affected request until the request is successful, is interrupted by an external source, or a configurable time-out period elapses.

Note: You must issue the commands described in this chapter at a console or terminal of the machine being configured; you cannot specify a different machine name to be used with these commands. Some of the commands require that you log in as **root**, while others require no privileges; the necessary privileges are indicated with each command. All of the files mentioned in this chapter are local files.

Choosing Cache Type, Location, and Size

The default DFS configuration is disk caching. However, the Cache Manager can use a machine memory cache rather than a disk cache. To direct the Cache Manager to use memory caching, use the **-memcache** option with the **dfsd** command. When the **-memcache** option is used, the Cache Manager does no disk caching, even if the machine has a disk available.

The **CacheInfo** file defines the directory to use for a disk cache and the size of a disk or memory cache. The Cache Manager checks this file at initialization to determine this information. The **CacheInfo** file contains the following three fields separated by colons:

- A directory on the local disk where the Cache Manager mounts the DCE global namespace. The default entry is the global namespace designation (**/...**). If **/...** is not specified, symbolic links to the global namespace fail.
- A local directory that serves as the DFS cache for a disk cache. The Cache Manager creates its cache files in this directory. The default entry is */dcelocal***/var/adm/dfs/cache**. Although the indicated directory is not used with a memory cache, an entry *must* appear in this field even if memory caching is employed on the machine. On AIX, create a separate AIX JFS file system and mount it at **/dcelocal/var/adm/dfs/cache** for the on-disk cache.
- A definition of the cache size in kilobyte blocks.

Following is an example of a **CacheInfo** file. The file lists the DCE namespace mounted at the global namespace designation (**/...**), the */dcelocal***/var/adm/dfs/cache** directory used for the cache, and a defined cache size of 50,000 kilobyte blocks (the machine *must* have this many blocks available on its disk):

```
/...:dcelocal/var/adm/dfs/cache:50000
```

Altering Default Parameters with the dfsd Process

The fields in the **CacheInfo** file are the only Cache Manager parameters that you *must* set. However, you can use the options available with the **dfsd** command to alter several Cache Manager defaults, affecting the way information is cached. Following are the configuration parameters that have the greatest effect on cache performance and security. (See the following subsections for a description of the **dfsd** options used to configure these parameters.)

- Total Cache Size: This is the amount of disk space or memory available for caching.
- Chunk Size: This parameter determines the maximum amount of data that can fit in a cache chunk. A chunk cannot hold more than one element; in a memory cache, the unused memory that is allocated for a chunk is wasted. If an element cannot fit in a single chunk, it is split into as many chunks as are needed.

  This parameter also determines the maximum amount of data that the Cache Manager can request at one time from a File Exporter. Increase the chunk size to take advantage of very fast links or decrease the size for slow networks.
- Cache Chunk Configuration: This parameter determines the number of chunks that are used for the cache. It can affect how often the Cache Manager must discard cached data to make room for new data. Consider the following example:

  A disk cache is configured at 50 megabytes and consists of 1000 chunks. Suppose 10 users each have the Cache Manager cache 100 files and each file is 20 kilobytes in size. This uses all 1000 chunks available (because each chunk can hold only one element), even though the cache has only 20 megabytes of cached elements, which is less than 50% of its capacity of 50 megabytes.

  When a user requests more data, the Cache Manager must discard cached data to reclaim space, even though the cache is not close to its capacity. In this case, increasing the number of chunks into which the cache is divided would improve performance by allowing the unused 30 megabytes of cache capacity to be allocated for other cached files.

- Number of dcache (disk cache) Entries in Memory: The Cache Manager maintains one dcache entry for each cache chunk; the entry records system identification information such as the file ID and version number of the file corresponding to the chunk. On disk caching machines, each dcache entry is stored in the **CacheItems** file, with a small number of entries (by default, 100) duplicated in machine memory. On memory caching machines, all dcache entries are stored in memory; the number of entries is equal to the number of chunks.
- Minimum and Initial RPC Authentication Levels: The Cache Manager maintains an initial authentication level used as a starting point for authentication negotiations with File Servers and a minimum acceptable level for communications with File Servers. There are two sets of these parameters: one set governs communications with File Servers in the local cell, and the other set governs communications with File Servers in foreign cells.

Note: The **dfsd** command is normally placed in a machine's initialization file (**/etc/rc** or its equivalent), not typed at the console. The commands in the following subsections are presented as examples of entries from initialization files.

Disk Cache Configuration

The number of kilobyte blocks allocated to the cache is defined in the third field of the **CacheInfo** file. You can use the **-blocks** option to override the number of cache blocks; the unit of measure associated with the cache block size is always kilobytes. The Cache Manager heuristically divides the number of blocks by 8 to determine the number of cache chunks in a disk cache. The following example sets the number of disk blocks allocated for the cache to 75,000 kilobyte blocks:

```
dfsd -blocks 75000
```

The default number of cache chunks in a disk cache is computed as the number of cache blocks divided by 8; you can use the **-files** option with a positive integer not greater than 32,000 to override this default. To use your cache most effectively, issue the **du** command on the cache directory to determine the number of cache blocks used; compare this number to the number of blocks allocated to the cache. If you are not using 85 of the cache, increase the number of chunks (**V** files). The following example sets the number of chunks to 2000:

```
dfsd -files 2000
```

The default chunk size for a disk cache is 64 kilobytes (2^{16}); the unit of measure associated with the chunk size is always bytes. You can use the -chunksize option to override the default chunk size. Provide an integer between 13 and 18 to be used as an exponent of 2. For example, a value of 15 sets the chunk size to 32 kilobytes (2^{15}= 32,768); a value of 16 equals the default for disk caches (2^{16}= 64 kilobytes). A value less than 13 or greater than 18 returns the chunk size to the default (as does a value of 16). The following example sets the chunk size to 16 kilobytes (2^{14}):

```
dfsd -chunksize 14
```

For a disk cache, the default number of dcache entries duplicated in memory is 100. You can use the **-dcache** option with a positive integer to change the default. It is usually not necessary to duplicate more than 100 entries in memory. However, because memory access is faster than disk access, increasing the number of dcache entries stored in memory may improve performance slightly. The following example sets the number to 250:

```
dfsd -dcache 250
```

When altering a disk cache configuration, any combination of **dfsd** options is allowed. However, the cache size defined in the **CacheInfo** file or with the **-blocks** option cannot be exceeded with the **-files** or **-chunksize** option.

Memory Cache Configuration

The default chunk size for a memory cache is 8 kilobytes (2^{13}). There is no predefined default for the number of chunks in a memory cache, and, as mentioned previously, the number of dcache entries equals the number of chunks.

If the **-blocks** option is used alone, it overrides the default cache size in the **CacheInfo** file. The Cache Manager divides this value by the default chunk size of 8 kilobytes to calculate the number of chunks and dcache entries. The following example sets the cache size to 5 megabytes (5120 kilobytes); as a result, the number of chunks is set to 640 (5120 divided by 8):

```
dfsd -memcache -blocks 5120
```

If the -chunksize option is used alone, it overrides the default of 8 kilobytes (2^{13}). Provide an integer between 13 and 18 to be used as an exponent of 2. For example, a value of 15 sets the chunk size to 32 kilobytes (2^{15}= 32,768). A value less than 13 or greater than 18 returns the chunk size to the default (as does a value of 13). The following example sets the chunk size to 16 kilobytes (2^{14}); if the total cache size is 4 megabytes (2^{12} kilobytes), the resulting number of chunks is 256:

```
dfsd -memcache -chunksize 14
```

If the -blocks and -chunksize options are used together, they override the defaults for the cache size and the chunk size. The Cache Manager divides the cache size by the chunk size to calculate the number of chunks and dcache entries. The following example sets the cache size to 8 megabytes (8192 kilobytes) and the chunk size to 16 kilobytes (2^{14}), resulting in 512 chunks:

```
dfsd -memcache -blocks 8192 -chunksize 14
```

When configuring a memory cache, the following options explicitly set the number of chunks and dcache entries. They also set the cache size indirectly and should not be used; use **-blocks**, **-chunksize**, or both, and allow the Cache Manager to determine the number of chunks and dcache entries itself.

- The **-dcache** option alone. The Cache Manager multiplies this value by the default chunk size (8 kilobytes) to derive a total cache size, overriding the value in the **CacheInfo** file.
- A combination of **-dcache** and **-chunksize** options. The Cache Manager sets the specified values and multiplies them together to obtain a total cache size, again overriding the value in the **CacheInfo** file.

Do not use the following options when configuring a memory cache:

- The **-files** option alone. This option sets the number of chunks for a disk cache and is thus ignored for a memory cache.
- The **-blocks** and **-dcache** options together. If you combine these options, the Cache Manager may choose one of the values to ignore, or the command may fail.

(See the Reference part of this guide and reference for complete information about these options and the use of the **dfsd** command in general.)

Changing Cache Location

The default directory for the Cache Manager's cache is *dcelocal***/var/adm/dfs/cache**. You can change this to a directory on another partition if more space is available. Use the **du** or **df** command (or an equivalent command) to check partition size and fullness.

You can change the location of the cache by editing the **CacheInfo** file or by using the **-cachedir** option with the **dfsd** command. (See the Reference part of this guide and reference for more information about the use of the **dfsd** command.)

To change the cache location, do the following:

1. Log in as **root** on the machine.
2. Use a text editor to change the second field of the **CacheInfo** file (the information between the two colons). The new directory must be on the local disk of the machine. The following example shows the cache directory specified in the **CacheInfo** file changed to **/usr/cache**:

   ```
   /...:/usr/cache:15000
   ```

3. Reboot the machine using **/usr/sbin/shutdown**or its equivalent. (Consult your system documentation for information on the correct rebooting command for your workstation.)
4. Move to the old cache directory and delete it using the **rm -rf** command.

Listing and Setting Cache Size

The amount of local disk space or memory allocated for the Cache Manager to use for its cache affects the speed of file access. A larger cache means the Cache Manager has to contact the File Server machine less often, resulting in fewer cross-network messages. A smaller cache fills sooner, making it more likely that the Cache Manager must discard cached copies of data to make room for newly requested data; if the user requests the discarded data again, the Cache Manager must recontact the File Server machine and refetch the data. A larger cache can make the initial discarding of data unnecessary.

The amount of disk space or memory used for caching depends on several factors. The size of the partition that houses the cache directory or the amount of memory available on the machine places an absolute limit on cache size. Do not use more than 85 of the cache directory's partition for a disk cache; do not use more than 20 to 25% of available memory for a memory cache (this leaves enough memory for processes and applications to run).

Within these limits, devoting more than 40 megabytes to the cache on a machine that does not serve multiple users is normally not useful unless users often work with large amounts of data (accessing large databases, for example). If a machine serves multiple users, a cache of 60 to 70 megabytes may be appropriate. A cache smaller than 5 megabytes can hamper Cache Manager performance; a cache smaller than twice the chunk size is rounded up.

You can reset the cache size for both types of caches by using a text editor to alter the size field in the **CacheInfo** file and then rebooting the machine; you must be logged in as **root** or have the write permission on the file to edit it. The **-blocks**

option can also be used with the **dfsd** command to override the **CacheInfo** value at reboot. (See the Reference part of this guide and reference for complete information about the **dfsd** command.)

To alter disk cache size without rebooting the machine, use the **cm setcachesize** command. The value remains in effect until the machine is next rebooted and reads the value in the **CacheInfo** file. To display the current cache size, the amount being used, and the type of cache (disk or memory), use the **cm getcachesize** command.

You must reboot to reset the cache size for a memory-caching machine.

Displaying the Cache Size from the CacheInfo File

Use the **cat** command (or the command appropriate to your system) to view the **CacheInfo** file on a client machine:

```
$ cat CacheInfo
```

The **CacheInfo** file contains a single line that lists three fields separated by a colon; the third field lists the maximum number of kilobyte blocks the Cache Manager can reserve for use as a cache in the designated cache directory.

In the following example, the default cache size for the machine is 25,000 kilobyte blocks:

```
$ cat CacheInfo
/...:dcelocal/var/adm/dfs/cache:25000
```

Displaying the Current Cache Size and the Amount in Use

Issue the **cm getcachesize** command on a client machine to view the current size of the cache and the amount in use. On machines that use disk caching, the current cache size may disagree with the default size specified in the **CacheInfo** file if the cache size was changed with the **cm setcachesize** command. Regardless of the type of caching in use, the current cache size may also disagree with the **CacheInfo** file if the size was changed with the **-blocks** option of the **dfsd** command.

```
$ cm getcachesize
```

The following example shows the number of kilobyte blocks that the Cache Manager is using as a cache at the moment the command is issued and the current size of the cache:

```
$ cm getcachesize
Using 13709 of the cache's available 25000 1K byte blocks.
```

Changing the Cache Size Temporarily

You can reset the cache size without rebooting a machine. The value remains in effect until you next reboot the machine. To change the cache size temporarily, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setcachesize** command to set a new cache size:

   ```
   # cm setcachesize -size kilobytes
   ```

The **-size** *kilobytes* option is the number of kilobyte blocks to be used for the cache. A value of 1024 equals 1 megabyte; the smallest allowable value is 1. A value less than 5120 (5 megabytes) can have a negative effect on Cache Manager performance; a value less than twice the chunk size is rounded up. A value of 0 (zero) resets the cache size to the amount specified in the **CacheInfo** file.

Resetting the Cache Size to the Default

To reset the cache size to the default, do the following:

1. Log in as **root** on the machine.
2. Use the **-reset** option with the **cm setcachesize** command to reset the cache size to the value specified at the last reboot, which is either the value in the **CacheInfo** file or the value set with the **-blocks** option of the **dfsd** command:

   ```
   # cm setcachesize -reset
   ```

   The **-reset** option resets the cache size to the value at the last reboot.

Changing the Cache Size Permanently

To change the cache size permanently, do the following:

1. Log in as **root** on the machine.
2. Use a text editor to change the number in the third field of the **CacheInfo** file. Specify a number in kilobyte blocks; 1024 kilobyte blocks equals 1 megabyte. Following is an example of the **CacheInfo** file:

   ```
   /...:dceloca/var/adm/dfs/cache:25000
   ```

   CAUTION:
   Be precise when editing the CacheInfo file. Use colons to separate the fields in the file; do not include any spaces in the file.
3. Reboot using **/etc/reboot** or its equivalent. (Consult your system documentation for information on the correct rebooting command for your workstation.)

Setting File Server and Fileset Location Server Machine Preferences

A replicated DCE LFS fileset typically has multiple read-only replicas. Each replica provides the same data, but each resides on a different File Server. When the Cache Manager needs to access data from a read-only replica, the FL Server provides the Cache Manager with the names of all File Servers on which that replica resides. As with File Servers, there can be multiple FL servers. The Cache Manager must choose the FL server to query for fileset locations and then choose the Fileset Server to access.

To choose from among these servers, the Cache Manager consults its collection of File Server and FL server preferences. Each preference consists of the hostname or Internet Protocol (IP) address of a server and the machine's numerical rank in the range from 1 to 65,534. The Cache Manager attempts to access the server that has the lowest recorded rank. If two FL servers have the same rank, the Cache Manager selects them in the order in which their names appear in the file system junction. If two File Servers have the same rank, the Cache Manager selects them in the order in which it received their names from the FL Server.

If the Cache Manager cannot access the server with the lowest rank—because the machine is currently unavailable, for example—it attempts to access the server with the next-lowest rank. It continues in this fashion until it either succeeds or determines that all File Servers or FL Servers are unavailable.

The Cache Manager stores its server preferences in the kernel of the local machine. Therefore, it loses its current preferences each time it is initialized. To rebuild its preferences following initialization, the Cache Manager assigns a default rank to each File Server or FL Server that it contacts. The Cache Manager bases its default ranks on IP addresses, as follows:

- If the local machine is also a File Server or FL Server, it receives an initial rank of 5000.
- Each File Server or FL Server in the same subnetwork as the local machine receives an initial rank of 20,000.
- Each File Server or FL Server in the same network as the local machine receives an initial rank of 30,000.
- Each File Server or FL Server in a different network from the local machine, or for which the Cache Manager can determine no network information, receives an initial rank of 40,000.

When multiple server entries have the same rank, the Cache Manager chooses between connections to suitable servers by picking the machine with the lowest ″round-trip time.″ When initialized, the Cache Manager assigns a randomly adjusted round-trip time value to each server entry in the preference list. The assigned round-trip time value is always greater than the limit for recorded values for actual round-trip times, thus biasing the selection process to any connection that the Cache Manager has actually made. The assigned round-trip time value ensures that if more than one entry have the same rank (a very probable event given the default values assigned to entries) then no single entry will receive all requests from the various DFS clients.

To display the Cache Manager's current set of File Server connection preferences, use the **cm getpreferences** command. Use the **cm getpreferences** command with the **-fldb** option to display the current set of FL Server connection preferences. By default, the command displays its output on standard output, but you can direct the output to a specified file. Note that each server will normally have up to four entries, each with a different IP address.

To specify preferences for one or more File Server connections, use the **cm setpreferences** command. Use the **cm setpreferences** command with the **-fldb** option to specify preferences for one or more FL Server connections. Each preference entry that you specify must consist of the following pair of values: the machine specification (either the host name or IP address of the File Server or FL Server) and that machine's rank in the form of an integer in the range from 1 to 65,534 (lower ranks have a higher preference). You can specify up to four preference entries for the same server, with each entry using a different IP address.

Use the command's options to specify File Server or FL Server preference entries as follows:

- Use the **-server** option to specify one or more File Server entries on the command line. If the **-fldb** option is not used, the **-server** option specifies File Server machine specifications and their ranks. For example, the following command assigns the File Server host names **fs1.abc.com** and **fs2.abc.com** ranks of 19,000 and 21,000, respectively:

  ```
  # cm setp -se fs1.abc.com 19000 fs2.abc.com 21000
  ```

- Use the **-fldb** option with the **-server** option to set one or more FL Server preference entries. For example, the following command assigns the FL Server host names **fl1.abc.com** and **fl2.abc.com** with ranks of 15,000 and 25,000, respectively:

  ```
  # cm setp -se fl1.abc.com 15000 fl2.abc.com 25000 -fldb
  ```

- Use the **-path** option to specify the pathname of a file that contains server preferences. (The output from the **cm getpreferences** command can be redirected to create such a file.) For example, the following command reads a collection of preferences from a file that resides on the local machine at **/etc/cm.prefs**:

  ```
  # cm setp -pa  /etc/cm.prefs
  ```

  The preference file should contain lines similar to the following:

  ```
  121.86.33.41 39000
  121.86.33.34 39000
  121.86.33.36 41000
  121.86.33.37 41000
  ```

- Use the **-stdin** option to read preference entries from standard input. For example, the following command reads preferences piped to the command from a user-defined program named **mkprefs**:

  ```
  # mkprefs | cm setp -st
  ```

  The program should generate preferences in the following format:

  ```
  fs3.abc.com 15000 fs4.abc.com 25000...
  ```

The **-server**, **-path**, and **-stdin** options are not mutually exclusive. You can include any combination of the options with the **cm setpreferences** command. If the Cache Manager already has a rank for a File Server or FL Server connection that you specify, the rank you specify replaces the connection's existing rank.

Each Cache Manager maintains its own collection of preferences, so two Cache Managers can have two different ranks for the same File Server or FL Server connection.

To load a predefined set of preferences each time the Cache Manager is initialized, include the **cm setpreferences** command in the machine's initialization file.

Displaying File Server and FL Server Preferences

Issue the **cm getpreferences** command to display the Cache Manager's preferences for File Servers or FL Server connections, as follows:

```
$ cm getpreferences [-path filename] [-numeric ] [-fldb ]
```

The **-path** *filename* option specifies a file to which the command is to write its output. Omit this option to display the preferences on standard output.

The **-numeric** option directs the command to display the IP addresses rather than the host names of the server machine connections. Omit this option to display the host names.

The **-fldb** option directs the command to display the FL Server machine connections and their respective ranks and not File Servers.

The command produces output of the following form for each server connection preference:

hostname rank

For example:

fl2.abc.com 25000

In the output, *hostname* is the host name of a File or FL Server, and *rank* is the machine's numeric preference rank. Note that *hostname* is replaced with the machine's IP address if the Cache Manager cannot presently determine the host name or if the **-numeric** option is included with the command.

Setting File Server Preferences

To set the Cache Manager's preferences for one or more File Server or FL Server connections, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setpreferences** command to set the Cache Manager's preferences for one or more File Server or FL Server connections, as follows:

   ```
   # cm setpreferences [-server machine rank...] [-path filename] \
   [-stdin ] [-fldb ]
   ```

   The **-server** *machine rank* option specifies one or more pairs of server machine specifications (either through the host names or IP addresses) and their ranks. By default, the machines are considered to be File Servers; however, you can specify FL Server entries by adding the **-fldb** option. Separate each machine specification and each rank with one or more spaces. Note that you can specify up to four entries per server, with each entry having a separate machine specification.

   The **-path** *filename* option specifies a file from which the command is to read one or more pairs of File Server specifications and their ranks. Separate each machine specification from its rank with one or more spaces, and include each paired machine specification and rank on a separate line.

   The **-stdin** option directs the command to read pairs of File Server specifications and their ranks from standard input. Separate each machine specification and each rank with one or more spaces.

   The **-fldb** option directs the command to set preferences for FL Server addresses, rather than for File Servers.

Determining setuid Permission

Programs that have **setuid** permission allow users to perform operations and access local files for which they normally may not have the necessary permissions. Such a program allows anyone who uses it to execute with the permissions of the user who owns the program for the duration of the program's execution.

While a **setuid** program executes, the person executing it is treated as the owner of the program. The effective user identification number (UID) of the executing program is the UID of the person who owns the program, not the UID of the person who initiated the program's execution. Thus, the person executing the program is granted the same permissions as the person who owns the program for as long as the program executes.

A **setuid** program owned by **root** allows a user who executes the program to execute with **root** privilege for the duration of the program. When handled correctly, such **setuid** programs are very useful. For example, programs that modify the password file for a system (**/etc/passwd** or its equivalent) are **setuid** programs that allow users to execute with **root** privilege long enough to modify their passwords. When handled incorrectly, however, **setuid** programs owned by **root** can present a serious breach in security.

In the UNIX operating system, **setuid** programs are indicated by setting a mode bit associated with a file. By default, the Cache Manager does not allow **setuid** programs to execute with **setuid** permission. Use the **cm setsetuid** command to enable **setuid** programs from specific filesets to execute with **setuid** permission. The command sets **setuid** status on a per-fileset and per-Cache Manager basis. It is commonly included in a start-up file (**/etc/rc** or its equivalent) to enable **setuid** programs from a specified fileset at machine startup.

Note that **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

Use the **cm getsetuid** command to determine whether the Cache Manager allows programs from specific filesets to execute with **setuid** permission.

Note: Every program also has a **setgid** bit that, when set, allows a person executing the program to execute with the permissions of the group that owns the program for the duration of its execution. When the **cm setsetuid** command is used, it automatically enables or disables **setgid** permission at the same time. Thus, if **setuid** programs are enabled on a fileset, **setgid** programs are also enabled on that same fileset.

Checking setuid Permission

Issue the **cm getsetuid** command to determine the status of **setuid** programs on specific filesets:

$ cm getsetuid [-path { *filename* | *directory_name*}...]

The **-path** option specifies a file or directory from each fileset whose **setuid** status is to be displayed. Omit this option to display the status for the fileset that contains the current working directory.

The output from this command includes a line for each specified fileset, stating one of the following:

- `no setuid allowed`

  Indicates that **setuid** (and **setgid**) programs from the fileset are disabled
- `setuid allowed`

  Indicates that **setuid** (and **setgid**) programs from the fileset are enabled
- `cm: the fileset on which '`*pathname*`' resides does not exist`

  Indicates that the pathname specified with the **-path** option is invalid

Changing setuid Permission

To change **setuid** permission, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setsetuid** command to change the status of **setuid** (and **setgid**) programs on specific filesets:

   ```
   # cm setsetuid [-path  {filename | directory_name}...] \
   [-state  {on | off}]
   ```

   The **-path** option specifies a file or directory from each fileset whose **setuid** status is to be changed. Omit this option to change the status for the fileset that contains the current working directory.

   The **-state on** option allows **setuid** programs from the indicated filesets to execute with **setuid** permission; the **-state off** option prevents **setuid** programs from the indicated filesets from executing with **setuid** permission. If this option is omitted, **setuid** programs from the specified filesets are allowed to execute with **setuid** permission.

Determining Device File Status

In UNIX file systems, devices are represented as special device files. By convention, device files reside in the **/dev** directory or a subdirectory of that directory; the UNIX kernel always honors device files stored in the **/dev** directory. However, the Cache Manager determines whether device files stored in filesets in the global namespace are honored. By default, the Cache Manager does not honor device files stored in filesets in the global namespace.

You can use the **cm setdevok** command to instruct the Cache Manager to honor device files stored on specific filesets. The command sets device file status on a per-fileset and per-Cache Manager basis. It is commonly included in a start-up file (**/etc/rc** or its equivalent) to honor device files at machine startup.

Use the **cm getdevok** command to determine whether the Cache Manager honors device files from specific filesets.

Checking Device File Status

Issue the **cm getdevok** command to determine whether the Cache Manager honors device files on specific filesets:

```
$ cm getdevok [-path  { filename | directory_name}...]
```

The **-path** option specifies a file or directory from each fileset about which device file status information is to be displayed. Omit this option to display the status for the fileset containing the current working directory.

The output from this command includes one line for each specified fileset, stating one of the following:

- `device files allowed`

  Indicates that device files from the fileset are honored
- `device files not allowed`

  Indicates that device files from the fileset are not honored
- `cm: the fileset on which 'pathname' resides does not exist`

Indicates that the pathname specified with the **-path** option is invalid

Changing Device File Status

To change device file status, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setdevok** command to change the status of device files on specific filesets:

   ```
   # cm setdevok [-path  {filename | directory_name}...] \
   [-state  {on | off}]
   ```

   The **-path** option specifies a file or directory from each fileset for which device file status is to be changed. Omit this option to change the status for the fileset containing the current working directory.

   The **-state on** option causes device files from the indicated filesets to be honored; the **-state off** option prevents device files from the indicated filesets from being honored. If this option is omitted, device files from the specified filesets are honored.

Updating Cached Data

When an application program requests new data and the cache is full, the Cache Manager discards some data to make room for the new information. It discards databased on the following two factors:

- If the data is reproducible; information is considered reproducible if it is unchanged from its first retrieval. By definition, data from read-only filesets is always reproducible; data from read/write filesets that was changed by a local application is considered reproducible if the changes are stored to the File Server machine.
- When the application program last referenced the data; data not used for the longest time is discarded first.

Thus, reproducible data not used for the longest time is discarded first. The Cache Manager continues to discard Least Recently Used (LRU) data in this fashion until there is enough room for the new data.

You can force the Cache Manager to discard, or flush, data cached from files, directories, and filesets. You can flush individual files or directories with the **cm flush** command, or you can flush one or more filesets with the **cm flushfileset** command. Flushing is necessary only in the event of file system problems or for testing purposes. The **cm flush** and **cm flushfileset** commands do not cause the Cache Manager to discard changes to data not written back to the central copies of files. These commands also do not affect data in the buffers of application programs.

The Cache Manager checks once an hour for changes that do not involve tokens, such as the release of a new version of a cached read-only fileset or a name change for any cached fileset. You can force the Cache Manager to notice these changes at other times with the **cm checkfilesets** command, which directs the Cache Manager to revise its table of mappings between fileset names and fileset ID numbers.

Flushing Specific Files or Directories

Issue the **cm flush** command to discard data from specific files or directories:

```
$ cm flush [-path  {filename | directory_name}...]
```

The **-path** option names each file or directory that you want to flush from the cache. If a *directory_name* is used, the Cache Manager flushes the name mappings and the blocks associated only with the directory, not with the files in the directory. If this option is omitted, the current working directory is flushed.

Flushing All Data from Specific Filesets

Issue the **cm flushfileset** command to discard data from specific filesets:

```
$ cm flushfileset [-path  {filename | directory_name}...]
```

The **-path** option names each file or directory in a fileset whose contents you want to flush. The Cache Manager flushes everything cached from each fileset that contains a specified file or directory. If this option is omitted, the fileset that contains the current working directory is flushed.

Forcing the Cache Manager to Notice Other Fileset Changes

Issue the **cm checkfilesets** command to make the Cache Manager check for changes to information about filesets that contain cached data:

```
$ cm checkfilesets
```

Discarding Unstored Data

The Cache Manager may occasionally be unable to write cached data back to a File Server machine, possibly because the File Server machine is down or because network problems prevent the Cache Manager from reaching it. In this event, the Cache Manager displays a message on the screen to notify the user that it cannot write the data to the File Server machine. If possible, it also returns a failure code to the application program that is using the data.

The Cache Manager keeps the unstored data in the cache and continues to attempt to contact the File Server machine until it can store the data. (The frequency with which the Cache Manager attempts to reach a File Server machine is defined with the **-pollinterval** option of the **fxd** command issued on that File Server machine.) In the meantime, corrective measures can be taken to alleviate the problem that prevents the data from being stored; for example, the File Server machine can be restarted. Once the problem is alleviated, the Cache Manager can contact the File Server machine and store the data.

The Cache Manager discards unstored data only when

- It needs to make room in the cache for other data. Given an average-sized cache with average usage, the Cache Manager rarely needs to discard unstored data.
- The **cm resetstores** command is issued to force the Cache Manager to discard unstored data from the cache. This command cancels the Cache Manager's continued attempts to contact unavailable File Server machines; *all* data that the Cache Manager cannot store to such File Server machines is discarded; you cannot selectively discard individual files or data from specific filesets.

The **cm resetstores** command affects only data that could not be written to a File Server machine; it does not affect other data in the cache. Nonetheless, issue the command only after issuing the **cm lsstores** command. The **cm lsstores** command lists the fileset ID numbers of filesets that contain data that the Cache Manager cannot write to a File Server machine. Examine the output of the command to be sure that you know from which filesets unstored data will be discarded. You may be able to use this information to ensure that unstored data from the indicated filesets can safely be discarded.

Note: Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

You can use the **cm statservers** command to determine which File Server machines are failing to respond to the Cache Manager. (See “Checking File Server Machine Status” for information about the **cm statservers** command.)

Listing Unstored Data

Issue the **cm lsstores** command to list filesets that contain unstored data that the Cache Manager cannot write back to a File Server machine:

```
$ cm lsstores
```

Discarding Unstored Data

To discard unstored data, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm resetstores** command to cancel any further attempts by the Cache Manager to contact unavailable File Server machines. The Cache Manager discards all data that it has been unable to store to such File Server machines.

   ```
   # cm resetstores
   ```

Checking File Server Machine Status

If the Cache Manager cannot access files or save files that it currently has cached, you can use the **cm statservers** command to determine if one or more File Server machines are currently inaccessible. The command checks the status of each File Server machine with which the Cache Manager has been in contact. The command does not report the reason that a File Server machine is unavailable, but it does list the names of unresponsive machines.

The Cache Manager classifies as unresponsive any File Server machine that meets the following pair of conditions:

- The Cache Manager has been in contact with the File Exporter running on the machine and needs to contact it in the future (for example, the Cache Manager is holding tokens to data on the machine).
- The File Exporter on the machine is not responding to the Cache Manager’s periodic probes (implying that it also is not responding to requests for data).

The Cache Manager does not probe all File Server machines in the local cell; it probes only those File Server machines that house data it has cached. Similarly, the **cm statservers** command is concerned only with File Server machines that do

not respond to the Cache Manager's probes. You can use the command to check the statuses of File Server machines that meet the first of the previous two conditions and reside in the local cell, in a specific foreign cell, or in any cell.

To check the status of each File Server machine with which the Cache Manager has been in contact, issue the **cm statservers** command:

```
$ cm statservers [{-cell cellname | -all }]
[-fast ]
```

The **-cell** *cellname* option specifies the name of a foreign cell whose File Server machines the Cache Manager is to check. The Cache Manager determines the status of each File Server machine with which it has been in contact from the specified cell. Use the **-cell** option or use the **-all** option to direct the Cache Manager to check File Server machines in all cells; omit both options to check the status of each File Server machine the Cache Manager has contacted from the local cell only.

The **-all** option directs the command to check the status of each File Server machine with which it has been in contact, regardless of the cell in which a machine resides. Use the **-all** option or use the **-cell** option to specify a specific foreign cell whose File Server machines the Cache Manager is to check; omit both options to check the status of each File Server machine the Cache Manager has contacted from the local cell only.

The **-fast** option directs the command to display the results of its most recent probes. The Cache Manager does not probe File Server machines to determine their statuses at the instant the command is issued.

If all of the File Server machines that it probes respond, the Cache Manager displays the following output in response to the command:

```
All servers are running.
```

If one or more of the File Server machines that it probes do not respond, the Cache Manager displays the following output:

```
These servers are still down: hostname
```

where *hostname* is the name of each File Server machine that fails to respond. In a multihomed server environment, the *hostname* corresponds to the machine name that the Cache Manager is currently using to access each File Server machine. The output does not contain multiple machine host names for the same File Server machine.

RPC Authentication Level Configuration

You can set the RPC authentication level for communications between the Cache Manager and File Servers. By default, such communications use the packet integrity DCE RPC authentication level (each RPC is authenticated and the data is checked to ensure that it was not modified in transit). However, circumstances at your site may require higher security or permit lower security for File Server communications. The following lists the range of authentication levels:

Default
: Use the DCE default authentication level.

None Perform no authentication.

Connect
Authenticate only when the Cache Manager establishes a connection with the File Server.

Call Authenticate only at the beginning of each RPC received.

Packet
Ensure that all data received is from the expected host (authenticate).

Packet Integrity
Ensure that all data received is from the expected host and verify that none of the data has been modified.

Packet Privacy
Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

Note: Higher authentication levels do incur some overhead and therefore cause some degradation in performance. Lower security levels, while more efficient, do carry an additional risk of attack.

You can set separate authentication levels for dealing with File Servers in the local cell and for dealing with File Servers in foreign cells. These authentication levels are set through two pairs of values. Each pair of values consists of the following:

- An initial RPC authentication level. This value sets the initial RPC authentication level used by the Cache Manager when it attempts to establish communications with a File Server. The initial level is used as a starting point in negotiating an RPC authentication level with the File Server.
- A minimum RPC authentication level. This value defines a lower bound RPC authentication level for the Cache Manager. Should a File Server request an authentication level below this level, the Cache Manager refuses communications with that File Server.

For a complete description of how the Cache Manager negotiates the RPC authentication level, see Section 2.5. In short, each File Server (File Exporter) maintains its own pairs of security values. These pairs set maximum and minimum bounds that control RPC authentication for communications with Cache Managers. As with the Cache Manager, one pair controls communications with Cache Managers within the local cell while the other controls communications with Cache Managers in foreign cells. By default, the RPC authentication settings at the File Server and Cache Manager negotiates to the packet integrity authentication level.

The Cache Manager begins the negotiation by sending an RPC to the File Server at the authentication level determined by its initial RPC setting. The File Manager then replies in one of the following three ways:

- The File Server accepts the RPC authentication level it received (the level fell within the range defined by its upper and lower bounds) and begins the process of sending and receiving fileset data with the Cache Manager.
- The File Server finds that the RPC authentication level is above its upper bound and sends a response to the Cache Manager instructing it to lower its authentication level. If the Cache Manager is currently using an authentication level equal to the Cache Manager's lower bound, the Cache Manager ceases attempts to communicate with the File Server.
- The File Server finds that the RPC authentication level is below its lower bound and sends a response to the Cache Manager instructing it to raise its authentication level.

The following sections detail how to initially configure the Cache Manager RPC authentication levels and how to adjust those levels for a running Cache Manager.

Configuring RPC Authentication Levels

The default Cache Manager and File Server authentication settings are such that they negotiates to the packet integrity authentication level. Use the following options to set the authentication levels at the Cache Manager:

- **-initiallocalprotectlevel** - Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the local cell.
- **-minlocalprotectlevel** - Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the local cell.
- **-initialremoteprotectlevel** - Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells.
- **-minremoteprotectlevel** - Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells.

Each of the above options takes either a string, abbreviated string, or integer value as an argument to define the RPC authentication level. The following lists the values you can use:

- **rpc_protect_level_default** or **default** or **0**: Use the DCE default authentication level.
- **rpc_protect_level_none** or **none** or **1**: Perform no authentication.
- **rpc_protect_level_connect** or **connect** or **2**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_protect_level_call** or **call** or **3**: Authenticate only at the beginning of each RPC received.
- **rpc_protect_level_pkt** or **pkt** or **4**: Ensure that all data received is from the expected host.
- **rpc_protect_level_pkt_integrity** or **pkt_integrity** or **5**: Authenticate and verify that none of the data transferred has been modified.
- **rpc_protect_level_pkt_privacy** or **pkt_privacy** or **6**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

The following example sets the initial RPC authentication level for the home cell to connect, the minimum authentication RPC level for the home cell to none, the initial RPC authentication level for foreign cells to packet privacy, and the minimum authentication level for foreign cells also to packet privacy.

```
$ dfsd -initiallocalprotectlevel rpc_protect_level_connect -minlocalprotectlevel

none -initialremoteprotectlevel 6 -initialremoteprotectlevel pkt_privacy
```

When configuring the authentication levels, any combination of `dfsd` options is allowed.

Changing the RPC Authentication Levels Temporarily

You can reset any of the RPC authentication levels without rebooting the machine by using the `cm setprotectlevels` command. The values you alter remain in effect until you reboot the machine.

To change the authentication levels temporarily:

1. Log in as **root** on the machine.
2. Issue the **cm setprotectlevels** command with the appropriate options.

```
# cm setprotectlevels [-initiallocalprotectlevel  level] \
 [-minlocalprotectlevel   level] \
 [-initialremoteprotectlevel   level] [-minremoteprotectlevel level]
```

The various options are defined as follows:

-initiallocalprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell.

-minlocalprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell.

-initialremoteprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells.

-minremoteprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells.

Where *level* is the appropriate RPC authentication level. The various levels can be set by specifying a complete string, an abbreviated string, or a corresponding integer value.

The example command does the following:

- Sets the initial authentication level for communications with File Servers in the local cell to packet integrity.
- Sets the minimum authentication level for communications with File Servers in the local cell to packet.
- Sets the initial authentication level for communications with File Servers in foreign cells to packet privacy.
- Sets the minimum authentication level for communications with File Servers in foreign cells to packet privacy.

```
$ cm setprotectlevels -initiallocalprotectlevel pkt_integ -minlocalprotectlevel
4 \
-initialremoteprotectlevel 6 -minremoteprotectlevel 6
```

Checking RPC Authentication Levels

You can check the Cache Manager's current RPC authentication levels by using the `cm getprotectlevels` command:

```
$ cm getprotectlevels
```

```
Initial protection level in the local cell: value
Minimum protection level in the local cell: value
Initial protection level in non-local cells: value
Minimum protection level in non-local cells: value
```

The various possible output strings for *value* are as follows:

- **rpc_c_protect_level_default** - default
- **rpc_c_protect_level_none** - none
- **rpc_c_protect_level_connect** - connect
- **rpc_c_protect_level_call** - call
- **rpc_c_protect_level_pkt** - packet
- **rpc_c_protect_level_pkt_integ** - packet integrity
- **rpc_c_protect_level_pkt_privacy** - packet privacy

Client Persistent Requests Configuration

By default, in the current DFS network model, there might be conditions encountered while making RPC calls to DFS file exporters or related services that result in error returns to applications accessing DFS data. Most applications fail as a result of these errors. However, there are environments where this is undesirable. Examples are environments with long running tasks, batch processing systems, or unattended tasks. Another example where Client Persistent Requests is beneficial is DFS in AIX HACMP environments where takeover intervals can exceed the current DFS time-out values for communication failures. Examples of events with lost access to DFS are network outages, machine failures, and loss of access to DCE core services.

In such cases, it is desirable to have the task block or wait for the unreachable file server resources to become available and then continue processing. Client Persistent Requests enable DFS to do this type of request handling.

Client Persistent Requests is configurable on a per-client basis at DFS start-up. Operations blocked due to outages are interruptible by a user or administrator with a **<Ctrl-c>** or a **kill** command against the blocked process. The cache coherency property of DFS governed by the DFS Token State Recovery (TSR) model is retained with persistent requests. In situations where cache coherency would be violated, an error is returned to the application per DFS TSR semantics. The DFS client's ability to fail over to alternate server sites for replicated filesets is also retained. Only after all replicas are exhausted can an operation be blocked. When one or several servers are unavailable, available servers remain accessible as long as resources allow.

Even with Client Persistent Requests enabled, some conditions return errors to applications. It is important that normal file system errors get returned to applications. Examples are EACCES, ENOSPC, and ENOENT. Many types of configuration or administrative errors still result in application error returns. Examples are failing to configure FLDB server entries in the CDS name space or allowing a DFS client cache file system to become full. The primary objective is to survive transient planned or unplanned hardware and software outages that temporarily prevent communications with DFS file exporters.

Configuring Client Persistent Requests

By default, the Cache Manager is not enabled for Client Persistent Requests. One way to enable this feature for a DFS client is at configuration time. Use **config.dfs** or the AIX SMIT utility. This ensures that **dfsd** is run with the **-persistentrequests** option each time it is started. Another way to enable persistent requests is by using the **cm setpersistreqs** command. See “Changing Client Persistent Requests Status” for more details.

Checking Client Persistent Requests Status

Issue the **cm getpersistreqs** command to determine the DFS client’s persistent requests status.

```
$ cm getpersistreqs [-help]
```

The **-help** option displays the command syntax. There are no other options with this command.

If Client Persistent Requests is turned off, the output message from the command is, ″DFS persistent requests processing is off.″

If Client Persistent Requests is turned on, the command outputs ″DFS persistent requests processing is on; timeout value = *xxxxx* seconds.″ *xxxxx* is the timeout value specified when Persistent Requests was turned on.

Changing Client Persistent Requests Status

Issue the **cm setpersistreqs** command to change the DFS client’s persistent requests status.

```
$ cm setpersistreqs  [-timeoutvalue numsecs] [-state {on | off}] [-help]
```

The **-timeoutvalue** option specifies the number of seconds that the process should be blocked and retried waiting for resources to become available. If this option is omitted when the **-state** *on* option is used, a default value of 86400 is assigned.

The **-state** *on* option turns persistent requests on for the client; the **-state** *off* option turns persistent requests off for the client.

The **-help** option displays the command syntax.

Chapter 9. Configuring the Backup System

The DFS Backup System can help you automate the process of making permanent copies of filesets on tape. You can create a full backup, which includes all of the data from every file in a fileset, or you can back up data incrementally, copying only those files that have changed since a previous dump. In the same fashion, you can restore filesets completely, or you can do an incremental, date-specific restore, which recreates the filesets as they were before a specific date.

Note: The DFS Backup System cannot be used with AIX JFS or AIX CD-ROM File Systems.

This chapter introduces the DFS Backup System. It describes configuration issues related to the performance of backup and restore operations. “Chapter 10. Backing Up and Restoring Data” on page 273 provides specific details about listing information from the Backup Database, backing up and restoring data, and administering the Backup Database. Refer to this chapter for information about configuring the Backup System and preparing it for backing up and restoring data; refer to “Chapter 10. Backing Up and Restoring Data” on page 273 for information about backing up and restoring data.

Introduction to the Backup System

With the DFS Backup System, you control many aspects of the backup process, including how often backups are performed, which filesets are backed up, and whether full or incremental backups are made. A dump or dump set is the result of performing a backup operation; it includes data from all of the filesets that were copied onto tape at the same time. A full dump includes data from every file in a fileset; an incremental dump includes only those files in the fileset that changed since a previous dump was made. The backup process is also referred to as *dumping a fileset family* or *creating a dump set*.

Once a fileset has been dumped, the DFS Backup System can be used to restore it. When restoring a fileset, the DFS Backup System first restores a full dump of the fileset. It then restores the changes to the fileset from any incremental dumps that were made since the full dump. Two types of restores are possible: a *full restore*, which recreates the fileset as it was at its last dump, including changes from the last full dump and any incremental dumps that were made since the last full dump; and a *date-specific restore*, which recreates the fileset as it was at the time of its last dump before an indicated date. The Backup System restores the last full dump and any incremental dumps of the fileset that were done before the specified date, so the fileset is current according to the last dump made before that date.

Sparse files contained in the backup fileset remain mostly sparse when dumped or restored. Any 64 KB chunk of a file that contains actual data expands to fill 64 KB on the disk. However, if a 64 KB chunk does not contain data it does not require physical space on the disk. This 64 KB granularity is imposed to ensure high-performance data access.

The Backup System can be used in conjunction with a variety of automated backup devices, including stackers and jukeboxes. Through a user-defined configuration file, you can specify parameters to configure the Backup System’s Tape Coordinator

to control automated backup equipment. The Tape Coordinator can then call executable routines, change tapes, select the proper tape, and handle errors.

The Backup Database records the schedule for backups, the locations of the Backup System's Tape Coordinators, the groups of filesets (fileset families) that can be dumped, and other administrative information. One Backup Database exists per cell; it is used to back up data from all administrative domains in the cell. A master copy of the Backup Database is maintained on one machine and replicated on other machines in the cell. Ubik creates and synchronizes the master and secondary copies of the database. (See "Chapter 2. DFS Configuration Issues" on page 27 for more information about Ubik.) In addition, the DFS Backup System provides facilities to back up the database by copying it to tape so that it can be restored if necessary. You can also remove specific configuration and dump information from the database if needed.

The Backup Database is maintained by the Backup Server, or **bakserver** process. The Backup Server must run on each machine that stores a copy of the Backup Database. Only the administrative users and members of the groups included in the **admin.bak** administrative list can issue **bak** commands, which are used to configure and administer the Backup System and to back up and restore data. Like the Backup Database, the **admin.bak** file is installed on one machine (usually the System Control machine) and copied to all machines that house copies of the Backup Database.

Some operating systems have their own backup commands. If your operating system has commands named **bak**, make certain that you use the complete pathname (*/dceshared*/**bin/bak**) when issuing DFS **bak** commands. Note that **bak** commands can presently be used to affect the Backup Database in the local cell only.

Tape Coordinator Machines

A Tape Coordinator machine is a machine on which backup and restore operations are physically conducted. To qualify as a Tape Coordinator machine, a machine

- Should be in a physically secure location.
- Must have one or more tape drives attached.
- Must be configured as at least a DCE client machine. Fewer configuration steps are required if the machine is also configured as some type of DFS server machine.
- Must be properly configured as a Tape Coordinator machine. For example, it must house the required configuration file, and it must have the necessary entries in the Backup Database.
- If you are using automated backup equipment (such as a stacker or jukebox), the Tape Coordinator machine must house the user-defined configuration file (which has the necessary parameters to control the specialized backup equipment).
- Must run one instance of the **butc** (BackUp Tape Coordinator or just Tape Coordinator) process for each tape drive.

The **butc** program must be active when you issue a **bak** command that involves either a tape drive or an operation being performed by a tape drive. For optimum efficiency, run several tape drives (and their Tape Coordinators). Start one **butc** process on a Tape Coordinator machine for each tape drive attached to the machine. Each Tape Coordinator controls the behavior of its associated drive and accepts service requests from the **bak** program.

A Tape Coordinator ID (TCID) identifies a Tape Coordinator; each TCID must be unique in the Backup System of the local cell. When you issue **bak** commands, specify a Tape Coordinator by specifying its TCID with the **-tcid** option. Depending on the command, the **bak** or **butc** program contacts one or more of the following: the Backup Database (by way of the Backup Server), the FLDB, or Fileset Server processes.

Fileset Families and Fileset Family Entries

When creating backups, you copy groups of filesets, known as *fileset families*, to tape. A fileset family includes all of the filesets that you want to dump together onto the same tape (or tapes, if the fileset family contains a large number of filesets). All of the filesets in a fileset family are dumped to tape with the same frequency (for example, once a day or once a week).

Fileset family entries (also referred to as *fileset entries*) define the filesets included in a fileset family. Each entry has three fields: the File Server machine name, the aggregate name, and the fileset name. Because filesets can be moved from File Server to File Server, the first two fields are usually designated with a **.*** wildcard, so a person backing up the filesets does not need to know the File Server machine and aggregate on which they reside. The last field, fileset name, is often specified with a regular expression pattern that matches certain fileset names. Within the Backup System, regular expression characters and the **.*** wildcard can be used in many arguments. (See “Defining Fileset Families and Fileset Family Entries” on page 261 for a description of these characters and their interpretations.)

With the Backup System, regular expression characters and the **.*** wildcard can be used in many arguments. (See “The MOUNT Parameter” on page 255 for a description of these characters and their interpretations.)

Dump Hierarchies and Dump Levels

A dump hierarchy is a logical structure that helps define the relationship between full and incremental dumps. As mentioned previously, an incremental dump includes only the files that changed since the fileset was last dumped; when creating an incremental dump, the Backup System uses a previous dump, known as the dump’s *parent*, to serve as a reference point on which to base the incremental dump.

A dump set is the product of dumping a fileset family at a certain dump level, which is an entry in the dump hierarchy. The dump set is a collection of data from filesets dumped at the same time and in the same manner (fully or incrementally). To create a dump set, you specify (with the **bak dump** command) both the fileset family and the dump level. The Backup System keeps all of the data in a dump set together on a tape (or set of tapes, if the dump set is too large to fit on a single tape). The name of the dump set consists of the name of the fileset family and the last component of the name of the dump level joined by a dot (*fileset_family_name***.** *dump_level*).

Each dump level can be associated with an expiration date that specifies when a tape containing data from that dump level can be overwritten. The expiration date is transferred to any backup tape that contains a dump made at that level. When dumping to tape, the system checks the tape for an expiration date. If the tape’s expiration date has not expired (if it is in the future), the system does not overwrite

the tape; if no expiration date is defined for the tape or if the tape's expiration date has expired (if it is in the past), the system overwrites the tape, but only with a dump set of the same name.

Command and Monitoring Windows

A single terminal session can be used to issue **bak** commands to the Tape Coordinators on all Tape Coordinator machines. The session corresponds to a command shell called the *command window*, which can be opened and closed without affecting the Tape Coordinators. This session can be run from any machine in the cell. Multiple command windows can be used, but they are not necessary.

A separate terminal session must be used for each Tape Coordinator and associated tape drive running on a machine; a terminal session of this type is referred to as a Tape Coordinator's *monitoring window*. The window must be a connection to the Tape Coordinator machine whose Tape Coordinator and tape drive it is monitoring. The Tape Coordinator runs in the foreground, so no further commands can be issued in the monitoring window. The monitoring window must remain open while the Tape Coordinator runs so that you can see the Tape Coordinator's prompts.

Note: When using automated backup equipment, such as a stacker or jukebox, you can set parameters in the user-defined configuration file to use default responses for all questions. This allows the automated backup equipment to run unattended. See "The MOUNT Parameter" on page 255 for more information.

Privileges Required to Use the Backup System

Three administrative lists, or **admin** files, determine the users who can perform specific backup and restore operations. Depending on the operation to be performed, you must be included in one or more of the following files:

- The **admin.bak** file on each server machine on which the Backup Database is stored. You must be listed in this file for any operation initiated by a **bak** command.
- The **admin.fl** file on each server machine on which the FLDB is stored. You must be included in this file for any operation that involves the Fileset Location Server (FL Server), such as backing up or restoring filesets.
- The **admin.ft** file on any File Server machine from which you dump filesets or to which you restore filesets. You must be listed in this file for any command that involves the Fileset Server, such as backing up or restoring filesets.

To avoid confusion, include any user who works with the Backup System on all three administrative lists on the appropriate machines. The operations in this chapter direct users to verify that they, meaning they or a group to which they belong, are included in the appropriate administrative lists (**admin.bak**, **admin.fl**, and **admin.ft**). Use the **bos lsadmin** command to display the members of an administrative list.

By default, the group **subsys/dce/dfs-admin** is added to all three administrative lists. Since **cell_admin** is a member of this group, a user DCE authenticated as **cell_admin** can perform **bak** command operations without any additional action. If you have users that need permission to perform certain **bak** command operations but you do not want to add them to the **subsys/dce/dfs-admin** group, you may

want to create a new DCE group and add these users as members. Then add this group to the appropriate administrative lists.

The user performing **bak** command operations must have write permission to the **/dcelocal/var/dfs/backup** directory.

Note: If you have not included users who are to issue **bak** commands on all three of these administrative lists, some commands may not work as detailed in this and the following chapter.

Permissions Needed to Run the butc Process

To start **butc**, the Tape Coordinator process, you must be DCE authenticated as **cell_admin**. To start the **butc** process from a DCE user other than **cell_admin**, run the following commands:

- **#cd /usr/lpp/dce/bin**
- **#chown root butc**
- **#chmod u+s butc**

You must be logged in as the local **root** user to run the preceding commands. These commands make the **butc** process a **setuid** program that runs as root so that it can access the **/krb5/v5srvtab** keytab file.

Permissions Needed to Run bak Commands

By default, the group **subsys/dce/dfs-admin** is added to all three admin lists, **admin.bak, admin.fl**, and **admin.ft**. Since **cell_admin** is a member of this group, a user DCE authenticated as **cell_admin** can perform **bak** command operations without any additional action being taken. If you have users that need permission to perform certain **bak** command operations but you do not want to add them to the **subsys/dce/dfs-admin** group, you may want to create a new DCE group and add these users as members. Then add this group to the appropriate admin lists. The user performing **bak** command operations must have write permission to the /**dcelocal/var/dfs/backup** directory.

Standard Information in this Chapter

The following subsections present standard options and arguments common to many of the commands described in this chapter. They also present some common operations repeated throughout this chapter.

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the Reference part of this guide and reference for complete details about each command.)

- The **-tapehost** *machine* option is the machine for which a Tape Coordinator is to be added. You can specify the machine's DCE pathname (for example, */.../abc.com/hosts/bak1*), its host name (for example, **bak1.abc.com**), or its IP address (for example, **11.22.33.44**).
- The **-family** *fileset_family_name* option is the name of the fileset family to be used in the command. The name must be unique within the Backup Database of

the local cell. It can be no longer than 31 characters. It can include any characters, but to avoid confusion when dump set names are created, it should not include a **.** (dot). Any regular expression characters entered on the shell command line must be escaped with a **** (backslash); for example, **usr*** for a fileset family named **usr***. To make it easier to track the contents of a fileset family, its name should give some indication of the contents of the fileset entries it contains; for example, use the name **user** for the fileset family that includes all user filesets in the file system.

- The **-level** *dump_level* option is the name of the dump level to be used in the command. The complete pathname of a dump level must always be specified. There are two types of dump levels:
  - Full dumps, which consist of a name preceded by a single **/** (slash) (for example, **/full**).
  - Incremental dumps, which consist of multiple elements that resemble a UNIX pathname listing the dump levels that serve as the parents of the dump level, starting with a full dump level and proceeding in order down the hierarchy; for example, **/full/weekly/monday**. An incremental dump level can consist of any number of elements; when defining a new dump level, all of the elements except the last one must already exist. Each level in a dump level name must be preceded by a **/** (slash).

  Dump levels should have meaningful names that give some indication of their purpose. A single element in a dump level name can be no longer than 28 characters, and the complete name can be no longer than 255 characters. Dump level names can include any characters, but to avoid confusion when dump set names are created, they should not include a **.** (dot). Regular expression characters included in a dump level name must be escaped with a **** (backslash) or ″ ″ (double quotes). The complete pathname of each dump level must be unique within the Backup Database of the local cell.
- The **-expires** *date* option is the expiration date for a dump level. Expiration dates can be specified in one of two ways:
  - Relative expiration dates, which use the keyword **in** to indicate a number of years, months, or days to be added to the current date to calculate the expiration date. When the system dumps a fileset at this level, it calculates the time at which the dump set expires by adding the values to the start time of the dump operation. Relative expiration dates are expressed as follows:

    in [*integer***y] [** *integer***m] [** *integer***d]**

    At least one value must be provided; multiple values must be listed in the order shown, with the appropriate unit abbreviation (**y**, **m**, or **d**) used with each value. For example, **in 1y 6m 2d** causes the system to add 1 year, 6 months, and 2 days to the current date to calculate the expiration date.
  - Absolute expiration dates, which use the keyword **at** to represent a specific date and, optionally, time to use as the expiration date. Absolute expiration dates are expressed as follows:

    at *mm*/*dd*/*yy* [*hh*:*mm*]

    If you specify a time, you must use 24-hour time. For example, **at 11/22/92 11:36** specifies an expiration date and time of 22 November 1992 at 11:36 a.m. If no time is provided, a default time of 00:00 (12:00 a.m.) on the indicated date is used.

  If you omit the **-expires** option from a command, tapes created at dump levels specified with the command have no expiration dates; they can be overwritten at

any time. Also, although the **-expires** options are followed by ellipses, you can specify only one expiration date. The ellipses are included only to accommodate the DFS command parser.

- The **-tcid** *tc_number* option is the TCID of the Tape Coordinator to be used for the command. Legal values are integers from 0 (zero) to 1023. If this option is omitted, the Tape Coordinator with a TCID of 0 is used to execute the command by default.

Standard Commands and Operations

The following subsections describe commands and operations that are used frequently in this chapter. If a command or operation is described in detail here, it generally is not described in depth in later sections of this chapter where it is used.

Starting a Tape Coordinator

Before performing a backup or restore operation, you must install at least one tape drive on a Tape Coordinator machine and define its corresponding Tape Coordinator in both the *dcelocal***/var/dfs/backup/TapeConfig** file and the Backup Database. (See “Configuring a Tape Coordinator Machine” on page 249 for a description of these and other configuration operations that must be performed.) This section explains how to start a Tape Coordinator. You must have a Tape Coordinator running any time you access a tape drive for use with the Backup System.

- If your tape drive supports compression, make sure that you turn off compression before executing the **fms** command.
- Make certain that you have the **w** (write) and **x** (execute) permissions on the *dcelocal***/var/dfs/backup** directory, which is the directory in which the Tape Coordinator creates its **TL** (log) and **TE** (error) files.
- Start a new terminal session on the Tape Coordinator machine to use as the monitoring window for the Tape Coordinator. It must remain open while the Tape Coordinator runs.
- In the newly opened window, issue the **butc** command to start the Tape Coordinator. The binary file for the **butc** program resides in the *dceshared***/bin** directory.

  ```
  $ butc [-tcid tc_number] [-debuglevel trace_level]
  ```

  The **-debuglevel** *trace_level* option specifies the type of messages to be displayed. There are two valid arguments:

  1 Indicates that the Tape Coordinator is to report on its activities as it restores filesets, in addition to prompting for new tapes as necessary.

  0 Indicates that the Tape Coordinator is only to prompt for new tapes; it also displays some output as necessary for operations that it executes. This is the default.

Note: If you are using an automated backup device, such as a stacker or jukebox, you can create a user-defined configuration file to control that device. This file is read by the **butc** command and is used to configure a Tape Coordinator. (See “The MOUNT Parameter” on page 255.for more information.)

Stopping a Tape Coordinator

When you are finished using a Tape Coordinator, you should stop it from running. To stop a Tape Coordinator process, enter an interrupt signal (**<Ctrl-c>** or its equivalent) in the Tape Coordinator's monitoring window.

Determining Tape Size and End-of-File Mark Size

The size of a tape determines the amount of data the Backup System can place on it. The tape size differs for different tape drives. In addition, the Backup System appends an end-of-file (EOF) mark after each fileset it dumps to tape. The size of the mark also affects the amount of space available for backup data on a tape. The values used for both of these figures are specified in the **TapeConfig** file once for each tape drive. Note that an EOF mark is appended after each fileset, not after each file.

If you do not know the tape capacity or EOF mark size for a tape drive, use the **fms** (file mark size) command to determine these values. The binary file for this command resides in the *dcesharedl***bin** directory. This command produces terminal output and an **FMSLog** file in the current directory; both the output and the **FMSLog** file list the tape capacity and the size of the EOF mark for the drive.

Note: Because this command inserts file marks onto the entire tape, it can take from several hours to more than a day to complete.

To determine the EOF mark size for a tape drive, do the following:

1. If your tape drive supports compression, make sure that you have turned off compression before executing the **fms** command.
2. Make certain you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions on the directory from which the command is issued. If the **FMSLog** file already exists in the directory, you need to have only the **w** permission on the file.
3. Insert a tape into the tape drive. The tape is overwritten while the command executes; you may want to use a blank tape or one that can be recycled.
4. Enter the **fms** command:

   $ fms -device *device_name*

   The **-device** *device_name* option specifies the name of the tape drive.

   An example of this command and its terminal output follows; the command also writes similar information to the **FMSLog** file. In the example, the tape size for the drive named **/dev/rmth1h** is 2,136,604,672 bytes; the EOF mark size for the drive is 1,910,220 bytes.

   ```
   $ fms /dev/rmth1h
   wrote block: 130408
   Finished data capacity test - rewinding
   wrote 1109 blocks, 1109 file marks
   Finished file mark test
   Tape capacity is 2136604672 bytes
   File marks are 1910220 bytes
   ```

Using the Interactive Interface

You can use the **bak** commands in regular command mode or in interactive mode. If you use interactive mode, note the following:

- You do not need to enter the string **bak** with each **bak** command; the `bak>` prompt replaces the command shell prompt.

- You do not have to escape regular expression characters; in regular command mode, you must place all regular expressions and wildcards in ″ ″ (double quotes) or escape each with a **** (backslash).
- You can track executing and pending operations with the **bak jobs** command; in regular command mode, you cannot track operations.
- You can cancel currently executing and pending operations with the **bak kill** command; in regular command mode, you cannot use the **bak kill** command.
- You do not have to establish a new connection each time you issue a command, so execution time is quicker; in regular command mode, each command establishes new connections to the **bakserver** and **flserver** processes, as necessary.

Most of the operations described in this chapter are presented in regular command mode. Where appropriate, some operations include steps introduced as *Optional* to indicate where working in interactive mode could be useful. The **bak jobs** and **bak kill** commands can be entered *only* in interactive mode.

Entering Interactive Mode: Enter the **bak** command:

```
$ bak
```

Leaving Interactive Mode: Enter the **quit** command at the bak> prompt:

```
bak> quit
```

Configuring the Backup System

Before using the Backup System for backing up and restoring data, you must ensure that certain conditions have been met. The following subsections explain in detail how to perform the following prerequisite tasks:

- Configuring Tape Coordinator machines
- Configuring a backup Backup System operator
- Defining fileset families and fileset family entries
- Defining a dump hierarchy of dump levels
- Labeling tapes, if necessary

See “Chapter 10. Backing Up and Restoring Data” on page 273 for information on inspecting the status of these prerequisites. If all of the prerequisites are met, turn to the appropriate section in “Chapter 10. Backing Up and Restoring Data” on page 273 for information on using the Backup System.

Configuring a Tape Coordinator Machine

Setting up a Tape Coordinator machine consists of using **bak** commands to configure the Tape Coordinators for the machine. You must also create the **TapeConfig** file, which includes a line for each Tape Coordinator on the machine. Each line defines the

- Size of tapes used in the drive, in kilobyte, megabyte, or gigabyte units.
- End-of-file (EOF) mark size for the drive. EOF marks are placed between filesets on a tape. The size of the EOF mark can differ for each type of tape drive.
- Device name (for example, **/dev/rst0**) of the drive.
- Tape Coordinator ID (TCID) of the drive.

Each tape drive and its Tape Coordinator must be assigned a TCID in the range from 0 to 1023; the TCID must be unique in the Backup Database of the local cell. When assigning the TCID, you must define the ID number in the Backup Database with the **bak addhost** command; you must also include the TCID in the entry for the tape drive in the **TapeConfig** file on the local disk of the Tape Coordinator machine.

Perform the following tasks once, when you initially configure a Tape Coordinator machine:

1. Prepare the tape drives.
2. Create the **TapeConfig** file that defines the tape parameters for each drive.
3. Create an entry in the Backup Database for the Tape Coordinator for each drive.

The following subsections describe the steps necessary to configure a machine as a Tape Coordinator machine. The instructions in the following section are required only if the machine to be configured as a Tape Coordinator machine is not a DFS server machine of some type (the machine must at least be a DCE client). The instructions in “Steps Required for All Machines” on page 251 are required for any machine to be configured as a Tape Coordinator machine.

Steps Required for a Client-Only Machine

You must perform the following steps to configure a DCE client that is not a DFS server machine of some type (for example, a File Server machine or a Backup Database machine) as a Tape Coordinator machine. For a client-only machine, perform these steps before you perform the steps in “Steps Required for All Machines” on page 251; perform the steps on the machine that is to be configured as a Tape Coordinator machine. Do not perform these steps for a machine that is configured as some type of DFS server machine.

1. Verify that the */dcelocal*/**var/dfs** and */dcelocal*/**var/dfs/backup** directories exist on the machine. Create the directories if they do not already exist.
2. Verify that you have the permissions necessary to create and modify principals and accounts in the Registry Database (for example, you need the **i** (insert) permission to create a principal in the **hosts/***dce_hostname* directory, where *dce_hostname* is the name of the machine as it was configured in DCE). If necessary, use the **dcecp acl show** command to determine your permissions for a directory.
3. Use the **dcecp principal create** command to create a DFS server principal for the client machine that is to be configured as a Tape Coordinator machine:

   $ dcecp

   dcecp> principal create hosts/ *dcehostname*/**dfs-server**

   In the command, *dce_hostname* is the DCE hostname of the machine to be configured as a Tape Coordinator machine (for example, **client1.austin.ibm.com**). The DFS server principal created in this step is used in all subsequent steps that require the DFS server principal of the machine. (Machines configured as some type of DFS server machine receive DFS server principals when they are configured.)
4. Use the **dcecp account create** command to create an account for the DFS server principal of the machine:

   ```
   dcecp> account create hosts/ dce_hostname/dfs-server \> \
   -group subsys/dce/dfs-admin -org none \> \
   -password acct_password -mypwd your_password
   ```

In the command, **hosts/** *dce_hostname***/dfs-server** is the DFS server principal for which an account is to be created. The remaining options provide the following information:

- The **-group** *subsys/dce/dfs-admin* option specifies that the primary group of the account is to be the group named **subsys/dce/dfs-admin**. (The DFS server principals of all machines configured as some type of DFS server machine are added to this group when the machines are configured.)
- The **-org** none option specifies that the organization of the account is to be the organization named **none**.
- The **-password** *acct_password* option provides the password for the account of the DFS server principal. Choose a string that you can remember. You use the **dcecp keytab add** command to generate a random password for the account later in these instructions, so you do not need to enter a complex password at this time.
- The **-mypwd** *your_password* option is your password (the password for the DCE account to which you are currently authenticated).

5. Use the **dcecp keytab add** command to add a server encryption key for the DFS server principal to the default local keytab file, **/krb5/v5srvtab**. The **dced** process recognizes the keytab file by the name **self**. The command creates the keytab file if the file does not already exist. Use the **-member** option to specify the name of the DFS server principal, and use the **-key** option to specify the password that you entered for the principal's account in the previous step.

   ```
   dcecp> keytab add self -member hosts/ dce_hostname/dfs-server \> \
   -key acct_password
   ```

6. Use the **dcecp keytab add** command to create a new server encryption key for the DFS server principal. Use the **-member** option to specify the name of the DFS server principal. The **-random** option directs the command to generate a random string for use as the principal's server encryption key, and the **-registry** option directs the command to update the password of the principal's account in the registry database to match the randomly generated encryption key.

   ```
   dcecp> keytab add self -member hosts/dce_hostname/dfs-server \> \
    -random -registry
   ```

7. Use the **dcecp acl modify** command with the **-add** option to add an entry for the group **subsys/dce/dfs-admin** to the ACL of the entry for the DFS server principal in the security namespace. The **-add** option provides the ACL entry to be added to the ACL of the principal's entry. The permissions included in the ACL entry allow members of the specified group to perform all required operations on the principal's entry.

   ```
   dcecp> acl modify /.../ cellname/sec/principal/hosts/dce_hostname/dfs-server \> \
   -add {group subsys/dce/dfs-admin rcDnfmag} dcecp> exit
   ```

Once you have completed these steps, perform all of the steps in "Steps Required for All Machines".

Steps Required for All Machines

You must perform the following steps to configure any machine as a Tape Coordinator machine. If the machine to be configured is a DCE client but not a DFS server machine of some type, you must perform all of the steps in "Steps Required for a Client-Only Machine" on page 250 before performing the steps in this section. If the machine is configured as some type of DFS server machine, you do not need to perform the steps in "Steps Required for a Client-Only Machine" on page 250.

1. Install one or more drives on the machine according to the manufacturer's instructions. The Backup System can track a maximum of 1024 drives in a cell.

2. Verify that you have the **w** (write) and **x** (execute) permissions on the *dcelocal***/var/dfs/backup** directory (the directory in which you must create the **TapeConfig** file).
3. Create the *dcelocal***/var/dfs/backup/TapeConfig** file on the machine with a text editor. Use a single line in the file for each tape drive attached to the Tape Coordinator machine, recording the following information:
   - The tape size of the tapes to be used in the drive. The Tape Coordinator uses this capacity as the size of all tapes used in the drive. It is recommended that you use a number 10 to 15% lower than the actual tape capacity to allow for tape variations. The following abbreviations can be used for the tape size unit of measurement (the default is kilobytes); do not leave a space between the number and the letter.
     – Kilobytes: k or K (for example, 2k or 2K)
     – Megabytes: m or M (for example, 2m or 2M)
     – Gigabytes: g or G (for example, 2g or 2G)
   - The EOF mark size for the type of tape to be used in the drive. The Backup System appends an EOF mark after each fileset dumped to tape. The size of this mark can affect the amount of space available for backup data. The EOF mark size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. It is recommended that you increase the actual file mark size by 10 to 15% to allow for tape variations.

     If you do not specify a unit of measurement, the default unit used for the EOF size is bytes (*not* kilobytes, as for tape capacity). To indicate other units, use the same abbreviations as for tape capacity.
   - The device name of the tape drive. The format of this name varies with each operating system. For example, in the UNIX operating system, a valid device name is **/dev/rst0**.
   - The TCID for the Tape Coordinator associated with the drive. The Backup System can track a maximum of 1024 tape drives; legal values are integers from 0 to 1023. You do not have to assign the numbers in sequence, and you can skip numbers. The TCID for any Tape Coordinator must be unique among all TCIDs in the local cell.

     Because the **bak** commands that require you to specify a TCID always use a default TCID of 0, assign a TCID of 0 to the Tape Coordinator for the drive that you will use most often; this enables you to omit the **-tcid** option as often as possible.

   If you do not know the tape size or the EOF mark size for the tape drive, determine them by using the **fms** command, as described in “Determining Tape Size and End-of-File Mark Size” on page 248.

   Following is an example listing of the contents of the **TapeConfig** file for a machine with two drives. The tape size for each drive is 2 gigabytes; the EOF mark size for each drive is 1 megabyte. The respective TCIDs of the two drives are 0 and 1.

   ```
   2g 1m /dev/rmth0h 0
   2G 1M /dev/rmth1h 1
   ```
4. Ensure that the **bak** and **butc** binary files are stored on the local machine. The **bak** file should be stored in the *dcelocal***/bin** directory; the **butc** file should be stored in the *dceshared***/bin** directory (a symbolic link to the file may exist from the *dcelocal***/bin** directory).
5. Verify that the individuals who are to use the Backup System are included in the appropriate administrative lists (see “Privileges Required to Use the Backup

System" on page 244); if necessary, issue the **bos lsadmin** command to check. You also need to ensure that you are included in the **admin.bak** list to issue the **bak addhost** command that follows. To add someone to a list, issue the **bos addadmin** command.

6. Verify that the **bakserver** process is running on the cell's Backup Database machines. If necessary, issue the **bos status** command to check.
7. *Optional*. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. The advantages of interactive mode are described in "Using the Interactive Interface" on page 276. The command in the following step assumes that regular command mode is used, not interactive mode.
8. Enter the **bak addhost** command to create an entry in the Backup Database for each Tape Coordinator, defining its TCID:

   ```
   $ bak addhost -tapehost machine [-tcid  tc_number]
   ```

   Repeat the **bak addhost** command for each Tape Coordinator to be added.

Configuring a backup Backup System Operator

You may want someone to run the Backup System who is not a **cell_admin**. Make the **butc** process a **setuid** program. This is done by changing the mode bits appropriately for **/usr/lpp/dce/bin/butc**.

1. Set up the backup system normally. If the Tape Coordinator machine runs no DFS servers, make sure to follow the additional client setup steps. The local user who performs backups must have `write` permission to the /*dcelocal***/var/dfs/backup** directory.
2. Make **butc** a setuid program that runs as **root** so that it accesses the keytab file, **/krb5/v5srvtab**. You must be logged in as local **root** to make this change.

   ```
   cd /usr/lpp/dce/bin
        chown root butc
        chmod u+s butc
   ```

3. Add the **t** (test) permission to the subsys/dce/dfs-bak-servers group's subsys/dce/dfs-admin ACL entry with **dcecp**.

   ```
   acl_edit /.:/sec/group/subsys/dce/dfs-bak-servrs
         sec_acl_edit> modify group:subsys/dce/dfs-admin:rctDnfmM
         sec_acl_edit> exit
   ```

4. Create a group in the registry, such as **bakadm**.

   ```
   rgy_edit
   rgy_edit=> domain group
   rgy_edit=> add bakadm
   rgy_edit=> quit
   ```

5. Create principals and registry with **dcecp**, for example, **pat**.

   ```
   rgy_edit
   rgy_edit=> domain principal
   rgy_edit=> add pat
   rgy_edit=> quit
   ```

6. Create accounts for the principals, giving membership in the **bakadm** group.

   ```
   rgy_edit
         rgy_edit=> domain account
         rgy_edit=> add pat
         Enter account group (gname": bakadm
         Enter account organization (oname": none
         Enter password:   patspass
         Retype password:   patspass
         Enter your password:   -dce-
         Enter misc info: () Pat
         Enter home directory: (/) /home/pat
   ```

```
Enter shell: () /bin/ksh
Password valid (y/n"? (y)
Enter expiration date Yyy/mm/dd or 'none'": (none)
Allow account to be server principal Yy/n"? (y)
Allow account to be client principal Yy/n"? (y)
Account valid for login Yy/n"? (y)
Allow account to obtain post-dated certificates Yy/n"? (n)
Allow account to obtain forwardable certificates Yy/n"? (y)
Allow certificates to this account to be issued via TGT authentication
  Yy/n"? (y)
Allow account to obtain renewable certificates Yy/n"? (y)
Allow account to obtain proxiable certificates Yy/n"? (n)
Allow account to obtain duplicate session keys Yy/n"? (n)
Good since date Yyy/mm/dd.hh:mm": (1997/05/31.11:04)
Create/Change auth policy for this acct Yy/n"? (n)
Add Account=> Enter account id (pname":
rgy_edit=> quit
```

7. Add the group to the admin.bak, admin.fl, and admin.ft admin lists with the **bos** command.

```
for list in bak fl ft; do
       bos addadmin -server /.:/hosts/<your_hostname> -adminlist
              admin.$list -group bakadm
      done
```

8. **dce_login** as **pat** to establish security contexts in the shells that are to run **butc** and **bak**.
9. Start **butc** and **bak** in the proper shells.

Creating a User-Defined Configuration File

You can create a user-defined configuration file to support automated backup equipment, such as stackers and jukeboxes. These devices automatically switch tapes during a dump. Jukeboxes can also automatically fetch the proper tapes for a restore operation. To handle the varying requirements of automated backup equipment, the user-defined configuration file calls executable routines that you create to operate your backup equipment. Through the user-defined configuration file, you can select the level of automation you want the Tape Coordinator to use.

Each backup device on a Tape Coordinator machine can have its own user-defined configuration file. The file must reside in the *dcelocal***/var/dfs/backup** directory and it must have a name in of the form **conf**_ *tape_device*, where *tape_device* specifies the relevant device. A separate file is required for each backup device.

When starting a Tape Coordinator, the **butc** program reads the **conf**_tape_device file and configures the Tape Coordinator based on the parameter settings it finds in the file. The configuration file parameters are the following:

MOUNT
Names a file that contains an executable routine. The routine can mount an automated backup device, such as a stacker or jukebox.

UNMOUNT
Names a file that contains an executable routine to perform tape unmount operations for an automated backup device.

ASK Can be used to force all Backup System prompts to accept the default answers rather than query the operator. This does not affect the initial prompt to mount the first tape. This parameter is useful for fully automating the backup process.

AUTOQUERY
Can be used to disable the initial Tape Coordinator prompt to mount the first tape. This parameter is also useful for fully automating the backup process.

NAME_CHECK
Can be set to prevent the Backup System from checking tape names.

FILE Can be used to direct the dump to tape or to a specified file.

The following sections define each of the parameters in detail. “Sample User-Defined Configuration Files” on page 257 contains annotated sample scripts that illustrate typical routines to control automated backup equipment.

The MOUNT Parameter

By default, the Backup System prompts the operator to mount a tape before opening the tape device file. However, the **MOUNT** parameter provides a mechanism to load a tape through an automated backup device. The **MOUNT** parameter takes an absolute pathname as an argument:

MOUNT / *pathname*

The specified file contains the executable routine to load the tape.

The following information is passed from the Backup System to the executable routine:

- The tape device pathname.
- The tape operation, which is selected by issuing one of the corresponding **bak** commands. The set of tape operations follows:
  - **dump**
  - **labeltape**
  - **readlabel**
  - **restore**
  - **restoredb**
  - **savedb**
  - **scantape**
- The number of times the tape has been requested. If an error occurs when the tape device is opened, this value is incremented by one and the executable routine is called again.
- The tape name. If no tape name is specified, **none** is passed to the executable routine.
- The dump ID. This is a unique indentification code assigned by the Backup System. If no dump ID is specified, **none** is passed to the executable routine.

Note: If you do not specify the **MOUNT** parameter, the Backup System prompts the operator to mount the first tape.

You can use the **AUTOQUERY** parameter to prevent the Backup System from prompting the operator to mount the first tape. “Defining a Dump Hierarchy of Dump Levels” on page 264 discusses this parameter.

If the executable routine returns an exit code of 0, the backup process continues. An exit code of 1 aborts the backup process. Any other exit code causes the backup process to prompt the operator for the correct tape.

To abort the **MOUNT** parameter routine, type an **a** (for abort) in the Tape Coordinator monitoring window. The process then aborts the executable routine and prompts the operator to mount the correct tape.

The UNMOUNT Parameter

Like the **MOUNT** parameter, the **UNMOUNT** parameter specifies a file that contains an executable routine. In this case, the executable routine is used to remove a tape from an automated backup device. The **UNMOUNT** parameter takes an absolute pathname as an argument:

UNMOUNT / *pathname*

The file specified by the **UNMOUNT** parameter is executed when the Backup System closes a tape device (whether the close operation succeeds or fails).

The Backup System passes the following information to the executable routine:

- The tape device file path and name.
- The tape operation, which in this case is **unmount**.

The ASK Parameter

The **ASK** parameter determines whether the Backup System should prompt the operator when an error occurs or simply use the default responses. The **ASK** parameter does not disable the initial prompt to mount a tape. The parameter takes the following arguments:

YES Enables operator prompts for all error cases. Not specifying the **ASK** parameter has the same result.

NO Disables operator prompts for all error cases and assumes the default responses.

The possible error conditions are:

- A **bak restore** operation fails to restore a volume. The **YES** argument causes the Backup System to ask whether the operator wishes to continue the restore operation. The **NO** argument continues the restore.
- A **bak dump** operation fails to dump a volume. The **YES** argument causes the Backup System to ask whether the dump for that volume should be retried, the volume should be omitted, or the dump operation should be aborted. The **NO** argument proceeds with the dump but omits the volume.
- A **bak scantape** operation cannot determine whether there is a next tape in the dump set. The **YES** argument causes the Backup System to ask whether there are more tapes to be dumped. The **NO** argument assumes that there are more tapes.
- A **bak labeltape** operation attempts to label a non-expired tape. The **YES** argument causes the Backup System to ask whether the operation should proceed. The **NO** argument does not label the tape.

The AUTOQUERY Parameter

The **AUTOQUERY** parameter disables the Backup System's initial prompt to mount a tape. Use the **AUTOQUERY** parameter in conjunction with the **ASK** parameter to disable all prompting from the Backup System. The **AUTOQUERY** parameter has the following arguments:

YES Enables the operator prompt for the first tape in the dump set. Not specifying the **AUTOQUERY** parameter provides the same result.

NO Disables the operator prompt for the first tape. A **NO** argument is similar to the **-noautoquery** option for the **butc** command.

The NAME_CHECK Parameter

The **NAME_CHECK** parameter prevents the Backup System from checking tape names on dump operations. The parameter has the following valid arguments:

YES Enables tape name checking. The Tape Coordinator verifies that the tape name is either **NULL** or the same name as the dump set. Not specifying the **NAME_CHECK** parameter provides the same result.

NO Disables tape name checking. Any non-expired tape is acceptable.

Disabling name checking is useful for recycling tapes without first relabeling them.

The FILE Parameter

The **FILE** parameter specifies whether the dump and restore operations will write to or read from a tape or a file. The parameter has the following valid arguments:

YES Dump and restore operations use a file. The target pathname is specified in the **/opt/dcelocal/var/dfs/backup/TapeConfig** file.

NO Dump and restore operations use a tape device. Not specifying the **FILE** parameter provides the same result.

Keep these points in mind when using this parameter:

- If the Tape Coordinator needs another file to continue an operation, it prompts the operator to mount another tape but then continues the operation using the pathname specified in the **/opt/dcelocal/var/dfs/backup/TapeConfig** file. A good practice is to specify a pathname that is a link to another file. If you must then provide another file name, you can take advantage of the prompt for a new tape to change the link to a new pathname.
- Do not specify the **YES** argument when using a tape device or the **NO** argument when using a file. Neither arrangement works.
- If you specify **YES**, all **ioctl** calls are removed. Data is still written in 16 KB blocks; however, the position of database records is not a filemark position (as is normal with tapes). Positioning to a volume is done directly with a **seek** call.

Sample User-Defined Configuration Files

The following sample **conf**_tape_device files show how you might structure configuration files for stackers, jukeboxes, or file dumps. They are examples only and are not recommendations.

There are two general considerations concerning the **conf**_tape_device files (these considerations are discussed in detail in section 9.3.2.1.), as follows:

1. The Backup System passes the following parameters to the **conf**_tape_device file:
   - The tape device pathname
   - The tape operation
   - The number of times the tape has been requested
   - The tape name

- The dump ID

2. The Backup System responds to exit codes from the **conf**_tape_device file in the following ways:

   Exit code 0
   Continue the backup process.

   Exit code 1
   Abort the backup process.

   Any other exit code
   Prompt the operator for the correct tape.

Sample conf_*tape_device* File for Stackers*:* The following is an example of a configuration file for dealing with stacker-type automated backup equipment:

```
AUTOQUERY NO
ASK YES
MOUNT /opt/backup/stacker0.1
NAME_CHECK NO
```

This file specifies the following:

- The **AUTOQUERY** parameter tells the Backup System not to prompt the operator to mount the first tape.
- The **ASK** parameter tells the Backup System to prompt the operator when an error occurs during the backup process.
- The **MOUNT** parameter tells the Backup System to call the file **/opt/backup/stacker0.1** and execute the routine in that file to initialize the stacker.
- The **NAME_CHECK** parameter tells the Backup System not to ensure that the name of the next tape in the stack matches the dump set name.

The previous example calls the **/opt/backup/stacker0.1** file to initialize the stacker and load a tape. The following is an example of the routine that might be contained in that file:

```
#! /bin/csh -f

set devicefile = $1
set operation  = $2
set trys       = $3
set tapename   = $4
set tapeid     = $5

set exit_continue    = 0
set exit_abort       = 1
set exit_interactive = 2

# ------------------------------------------------------

if (${trys} > 1) then
   echo "Too many tries"
   exit ${exit_interactive}
endif

if ((${operation} == "dump")       | \
    (${operation} == "savedb"))    then

   stCmd_NextTape ${devicefile}
   if (${status} != 0) exit ${exit_interactive}
   echo "Will continue"
   exit ${exit_continue}
endif
```

```
if ((${operation} == "labeltape")  | \
    (${operation} == "readlabel")) then
   echo "Will continue"
   exit ${exit_continue}
endif

echo "Prompt for tape"
exit ${exit_interactive}
```

This routine makes use of only two of the parameters passed to it by the Backup System: **trys** and **operation**. It is a good practice to watch the number of attempts and exit if it exceeds one (which implies that the stacker is out of tapes). Note that this routine calls **stCmd_NextTape** for **dump** or **savedb** operations; however, your file should call whatever routine is required to load the next tape for your stacker. Also note that the routine sets the appropriate exit code to prompt an operator to load a tape if either the stacker cannot load a tape or a restore operation is in process.

Sample conf_*tape_device* File for Jukeboxes*:* The following sample **conf**_tape_device file configures the Backup System to control a jukebox device, and includes an **UNMOUNT** parameter:

```
MOUNT /opt/backup/jukebox0.1
UNMOUNT /opt/backup/jukebox0.1
ASK NO
NAME_CHECK NO
```

This file specifies the following:

- The **MOUNT** parameter tells the Backup System to call the file **/opt/backup/jukebox0.1** and execute the routine in that file to mount a tape.
- When the Backup System closes a tape device, it calls the file **/opt/backup/jukebox0.1** and executes the routine to remove the tape from the jukebox.
- The **ASK** parameter tells the Backup System not to prompt the operator for the initial tape.
- The **NAME_CHECK** parameter tells the Backup System not to ensure that the name of the next tape in the stack matches the dump set name.

The following sample **conf**_tape_device file shows the use of the **trys** and **operation** parameters:

```
#! /bin/csh -f

set devicefile = $1
set operation  = $2
set trys       = $3
set tapename   = $4
set tapeid     = $5

set exit_continue    = 0
set exit_abort       = 1
set exit_interactive = 2

# ------------------------------------------------------

if (${trys} > 1) then
   echo "Too many trys"
   exit ${exit_interactive}
endif

if ((${operation} == "labeltape")  | \
```

```
    (${operation} == "readlabel")) then
   echo "Won't read or write a tape label"
   exit ${exit_abort}
endif

if ((${operation} == "unmount") then
   jbComd_UnMountTape $(devicefile)
   exit
endif
```

This routine makes use of two of the parameters passed to it by the Backup System: **trys** and **operation**. The **trys** parameter monitors the number of attempts to load a tape. If the number of attempt exceeds one, the jukebox is unable to load a tape and the routine will exit and return an exit code of 2 (which will cause the Backup System to prompt the operator to load a tape).

In this scenario, tape names are not checked before using a tape as part of a dump set (recall that the **NAMECHECK** parameter was set to **NO**, disabling tape name checking). Therefore, if the **labeltape** operations is attempted the routine echoes a message saying that these operations cannot be performed. The executable routine then returns an exit code of 1, which causes the Backup System to abort the operation.

If an **unmount** is executed, the routine calls the **jbComd_UnMountTap** function to remove the tape from the drive.

Sample conf_*tape_device* File for Dump to File*:* The following sample **conf**_tape_device file configures the Backup System to dump directly to a file.

```
MOUNT /opt/backup/file
FILE YES
```

This file specifies the following:

- The **MOUNT** parameter calls an executable routine in the **/opt/backup/file** file.
- The **FILE** parameter is set to **YES**, indicating that the information should be dumped directly to a file. The pathname for the target dump file is set in the *dcelocal***/var/dfs/backup/TapeConfig** file.

The following is an example of a routine that might be contained **opt/backup/file** that demonstrates how to configure the Backup System to handle dumps to a file.

```
#! /bin/csh -f

set devicefile = $1
set operation  = $2
set trys       = $3
set tapename   = $4
set tapeid     = $5

set exit_continue    = 0
set exit_abort       = 1
set exit_interactive = 2

# ------------------------------------------------------

if (${trys} > 1) then
   echo "Too many trys"
   exit ${exit_interactive}
endif

if (${operation} == "labeltape") then
   echo "Won't label a tape/file"
```

```
   exit ${exit_abort}
endif

if ((${operation} == "dump")       | \
    (${operation} == "restore")    | \
    (${operation} == "savedb")     | \
    (${operation} == "restoredb")) then

   /bin/rm -f ${devicefile}
   /bin/ln -s /hsm/${tapename}_${tapeid} ${devicefile}
   if (${status} != 0) exit ${exit_abort}
endif
exit ${exit_continue}
```

This routine makes use of two of the parameters passed to it by the Backup System: **trys** and **operation**. The **trys** parameter monitors the number of attempts to write to or read from a file. If the number of attempts exceeds one, the Backup System is unable to write to or read from the file specified in the *dcelocal***/var/dfs/backup/TapeConfig** file. The routine then exits and returns an exit code of 2 (which causes the Backup System to prompt the operator to load a tape). The operator can use this opportunity to change the name of the file specified in the *dcelocal***/var/dfs/backup/TapeConfig** file.

In this scenario, tape names are not checked before using a tape as part of a dump set (recall that the **NAMECHECK** parameter was set to **NO**, disabling tape name checking). Therefore, if the **labeltape** operation is attempted the sample routine echoes a message saying that these operations cannot be performed. The executable routine then returns an exit code of 1, which causes the Backup System to abort the operation.

A **dump**, **restore**, **savedb**, or **restoredb** operation links to a new file using the **tapename** and **tapeid** to build the file name. The **tapename** and **tapeid** are used so that **restore** operations can easily link to the proper file.

Defining Fileset Families and Fileset Family Entries

Before actually performing a backup, you must create one or more fileset families in the Backup Database. A fileset family defines the groups of filesets that are to be dumped together. Fileset family names can be no longer than 31 characters, and they can include any characters. Avoid using a dot in the name of a fileset family; when a dump set is transferred to tape, the fileset family name and the last component of the dump level name are automatically joined by a dot to form the name of the dump set.

In regular command (noninteractive) mode, characters from the regular expression character set used in the name of a fileset family must be escaped with a **** (backslash) to prevent the command shell from expanding them; for example, **usr*** for a fileset family named **usr***. Because they have no meaning in the name of a fileset family, regular expression characters are not recommended.

Once you define a fileset family, you must then define the fileset family entries in the family. Each fileset family entry is defined in terms of one or more filesets and the location of each fileset on a File Server machine and aggregate. Each fileset family entry consists of three fields, with each field separated by a space. Following are the legal values for each field in a fileset family entry:

- File Server Machine Name: The name of the File Server machine that houses the filesets. You can specify the machine's DCE pathname (for example,

/.../abc.com/hosts/fs1), its host name (for example, **fs1.abc.com**), or its IP address (for example, **11.22.33.44**). You can also use the special **.*** wildcard for this field; this wildcard matches all of the File Server machines in the cell.

- Aggregate Name: The device name or aggregate name of the aggregate on which the filesets reside. You can use the **.*** wildcard for this field; the wildcard matches all aggregate names.
- Fileset Name: The names of the filesets to be backed up. You can use the **.*** wildcard for this field to match all fileset names. The following regular expression characters can also be used in this field of an entry:
  - The ***** (asterisk) character matches any number of repetitions of the previous character.
  - The **[ ]** (brackets) characters around a list of characters match any single instance of the characters in the list but no other characters.
  - The ^ (circumflex) character as the first character in a bracketed set of characters matches any single character other than the characters that follow it in the list.
  - The **?** (question mark) character matches any single character or no character.
  - The **.** (dot) character matches any single character, but a character must be present.
  - The **** (backslash) character before any other regular expression character, including itself, matches the literal value of the character.

  In noninteractive mode, you must surround an entire string with ″ ″ (double quotes) if it contains regular expression characters or you must escape each regular expression character with a **** (backslash); for example, use ″**user\..*\.bak**″ or **user\\..*\\.bak** to indicate all of the filesets that begin with the prefix **user.** and end with the extension **.bak**. Otherwise, the command shell attempts to resolve the regular expression characters rather than pass them to the **bak** command interpreter for resolution. Note that the **.*** notation is interpreted as a single wildcard that must be surrounded with double quotes in noninteractive mode (″**.***″). Characters specified in regular expressions are case sensitive.

All fileset family names must be unique within the Backup Database of the local cell. Create and delete fileset families with the **bak addftfamily** and **bak rmftfamily** commands. Create and delete fileset family entries with the **bak addftentry** and **bak rmftentry** commands.

Suggestions for Creating Fileset Family Entries

The **bak addftentry** command has arguments that correspond to the three fields in a fileset family entry: **-server** for the File Server machine name field, **-aggregate** for the aggregate name field, and **-fileset** for the fileset name field. By combining these arguments in different ways, you can create fileset entries for different groupings of filesets. Table 9 summarizes some suggested groupings.

Table 9. Suggestions for Creating Fileset Family Entries

For Entries That Include:	Use:	For Example:
All filesets in the cell's file system	The wildcard for all three arguments	".*" ".*" ".*"

Table 9. Suggestions for Creating Fileset Family Entries (continued)

Every fileset on a specific File Server machine	Machine name with **-server** and wildcard for **-aggregate** and **-fileset**	`/.../abc.com/hosts/fs1 ".*" ".*"`
Filesets on aggregates of the same name	The aggregate name with **-aggregate** and the wildcard for -server and **-fileset**	`".*" /dev/lv01 ".*"`
Every fileset with a common string of letters (such as a `.backup` extension)	The wildcard for **-server** and **-aggregate** , and a character string/regular expression for **-fileset**	`".*" ".*" ".*\.backup"`
All filesets on an aggregate	The machine name with **-server**, the aggregate name with -aggregate, and the wildcard for **-fileset**	`/.../abc.com/hosts/fs2 /dev/lv02 ".*"`
Every fileset on each File Server machine's similarly named aggregate that includes a common string of letters in its name (such as a `user.` prefix)	The wildcard for **-server**, the aggregate name with **-aggregate**, and a character string/regular expression for **-fileset**	`".*" /dev/lv03 "user\..*"`
Every fileset on one aggregate with a common string of letters in its name (such as a `sys` prefix and a `.readonly` extension)	The machine name with **-server**, the aggregate name with -aggregate, and a character string/regular expression for **-fileset**	`/.../abc.com/hosts/fs3 /dev/lv04 "sys.*\.readonly"`

Include in a fileset family only those filesets that you wish to dump to the same tape at the same time (for example, weekly, or daily) and in the same manner (fully or incrementally).

The two main types of fileset families are those based on a common fileset location (File Server machine and aggregate) and those based on similar contents (as reflected by a fileset name). Because filesets can be moved between machines and aggregates, use name-based fileset family entries rather than location-based ones. For name-based entries, specify the **.*** wildcard for the **-server** and **-aggregate** arguments of the **bak addftentry** command.

Adding a Fileset Family to the Backup Database

To add a fileset family to the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak addftfamily** command to create the fileset family. The fileset family remains empty until you use the **bak addftentry** command to define entries in it.

   $ bak addftfamily -family *fileset_family_name*

Adding a Fileset Family Entry to a Fileset Family

To add a fileset family entry to a fileset family, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.

2. Define the entries in a fileset family that was previously created with the **bak addftfamily** command; the Backup System automatically assigns each entry an index number, starting with 1 for the first entry in each fileset family. This number is used if the fileset entry needs to be removed.

   ```
   $ bak addftentry -family fileset_family_name -server machine -aggregate  name \
   -fileset name
   ```

   The **-server** *machine* option is the File Server machine that houses the filesets to be included in the entry. Legal values for a specific machine are the machine's DCE pathname, the machine's host name, or the machine's IP address. You can also use the **.*** wildcard, which matches all machine names.

   The **-aggregate** *name* option is the device name or aggregate name of the aggregate that houses the filesets to be included in the entry. The **.*** wildcard can be used to match the names of all aggregates.

   The **-fileset** *name* option is the name of a fileset to be included in the entry. The **.*** wildcard or any of the regular expression characters described previously can be used to match the names of multiple filesets.

Deleting Fileset Families from the Backup Database

To delete a fileset family from the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak rmftfamily** command to delete each fileset family that you no longer need. It is not necessary to delete the fileset entries in each fileset family first; they are deleted automatically when the family is removed.

   ```
   $ bak rmftfamily -family fileset_family_name...
   ```

Deleting a Fileset Family Entry from a Fileset Family

To delete a fileset family entry from a fileset family, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak lsftfamilies** command to determine the index number of the entry that you want to delete. This is necessary only if the fileset family contains multiple entries, in which case this command is used to list the entries; if the family contains a single entry, the index is 1.

   ```
   $ bak lsftfamilies -family  fileset_family_name
   ```
3. Use the **bak rmftentry** command to delete the entry:

   ```
    $ bak rmftentry -family  fileset_family_name -entry  \
   fileset_entry_index
   ```

   The **-entry** *fileset_entry_index* option is the index for the entry that you want to delete.

Defining a Dump Hierarchy of Dump Levels

A dump hierarchy consists of one or more full dump levels and any incremental dump levels that you create with the **bak adddump** command. The dump levels define how fileset families are to be dumped; all fileset family entries in a fileset family are dumped at the same time and in the same way (fully or incrementally). A dump of a fileset family at a particular dump level produces a dump set. To create a dump set, specify the name of the fileset family and the level at which that family is

to be dumped when you initiate the dump with the **bak dump** command. (See “Chapter 10. Backing Up and Restoring Data” on page 273 for a description of the **bak dump** command.)

A dump hierarchy is defined by the dump levels it contains. The term *full dump level* refers to a dump level used when creating full dumps, the term *incremental dump level* refers to a dump level used when creating incremental dumps, and the term *parent dump level* refers to a dump level that serves as the reference point for an incremental dump level. Both full dump levels and incremental dump levels can serve as parent dump levels.

Each dump level in the hierarchy can be associated with an expiration date that specifies the date and time at which a tape that contains a dump set made at that level can be overwritten. Expiration dates are specified with the **bak adddump** or **bak setexp** command. A dump level’s expiration date is automatically placed on a tape that contains a dump made at that level to provide an extra level of protection against accidental erasure of the information on the tape.

Whenever a tape is used, the Backup System always checks to see whether the tape already contains a dump set. If the tape contains a dump set, the Backup System overwrites the tape only with a dump set of the same name. If the Backup System determines that it can overwrite the dump set, it then determines whether an expiration date exists on the tape; if no expiration date is associated with a tape or if the expiration date associated with a tape has expired, the system overwrites the dump set on the tape with a dump set of the same name. However, if the tape’s expiration date has not expired, the system refuses to overwrite the tape.

Following are some general issues to consider when building a dump hierarchy:

- A dump level can have any number of elements. The **/** (slash) is used as a metacharacter to separate different levels in the dump hierarchy. Regardless of its level in the dump hierarchy (full or incremental), each element in a dump level name must be preceded by a **/** (slash).
- Any characters can be included in a dump level name. Regular expression characters included in a name must be properly escaped with a **** (backslash) or ″ ″ (double quotes).
- Do not include a **.** (dot) in the name of a dump level. When a dump set is transferred to tape, the last component of the dump level name becomes part of the dump set name. The elements of the dump set name (the fileset family name and the last component of the dump level name) are joined by a dot. For example, if a fileset family named **sys** is dumped at the incremental dump level **/weekly/monday**, the dump set name is **sys.monday**.
- The maximum length for any single element in a dump level name is 28 characters. This does not include the / (slash) that precedes the element.
- The maximum length for the complete name of a dump level (full or incremental) is 255 characters. This includes any / (slashes) that are part of the name.
- A dump level is specified by its pathname. A level can share parents, but the level itself must have a unique name. Following are examples of different dump specifications:
  - The **/full** specification defines a full dump level.
  - The **/full/week1** specification defines an incremental dump level, **/week1**, with **/full** as its parent.
  - The **/full/week1/thursday** specification defines **/thursday** as a dump level that refers to **/week1** as its parent; **/week1** refers to **/full** as its parent.

- The dump level that you use as the parent for an incremental dump must already exist in the hierarchy when you define the incremental dump. The complete pathname of each dump level must be unique within the Backup Database of the local cell.
- A dump hierarchy can contain more than one full dump level; each level defines a separate subhierarchy in which you can create different relationships between the dump levels. The following two common methods are available to relate the incremental dumps in a subhierarchy to the full dump level and to one another:
  - Each incremental dump refers to the same full dump as its parent. With this method, the dump sets created at each of the incremental levels contain all of the files in the fileset family that changed since the family was last dumped at the full level.
  - Each incremental dump level (other than the first) refers to a preceding incremental dump level as its parent, rather than to the full dump level. With this method, each incremental dump includes only those files modified since a dump was last done at its parent level. When you restore files dumped in this fashion, however, you must access more tapes: the tape that contains the full dump and the tape for each incremental dump done afterward.

  The two types of hierarchies can be mixed within a single subhierarchy by setting some incremental dumps to refer to the full dump level as their parent and setting others to refer to preceding incremental levels.
- There is no implied relationship between a fileset family and a dump subhierarchy; you can dump any fileset family at any level in any subhierarchy. When dumping a fileset, *do not* alternate between incremental dumps from different subhierarchies. To dump a fileset according to a different subhierarchy, start at the full dump level.
- Use names in the hierarchy that correspond to real-world times; these can help you remember when to create dumps at the different levels. However, the Backup System does not automatically back up filesets according to the names assigned in the dump hierarchy; it does not interpret dump level names, nor does it automatically perform an incremental dump on Thursday simply because there is a dump level called **thursday**.

A few general guidelines for using a dump hierarchy follow:

- To set up a group of tapes for archiving, make certain that you use unique dump names; for example, **monday1**, **tuesday1**, **monday2**, or **tuesday2**.
- To recycle tapes, use dump levels with the same name; for example, **monday** or **tuesday**.
- To archive tapes and recycle them at a later time, simply perform backups with a new set of tapes; the old set of tapes can then be archived. This creates multiple entries for the dump in the Backup Database. To restore filesets with multiple entries in the database, use the correct dump date to restore the correct information.

The Backup System prevents you from using out-of-date configuration information. For example, if a user deletes a full dump level in a hierarchy, and another user tries to start an incremental backup based on that full dump level, the incremental backup fails. The second user must view the dump hierarchy with the **bak lsdumps** command. This command updates the hierarchy with the most recent changes and lets the user determine another dump level to use with the command. (See “Chapter 10. Backing Up and Restoring Data” on page 273 for more information on the **bak lsdumps** command.)

Examples of Dump Hierarchies

Following are examples of two possible dump hierarchies. Each hierarchy backs up data in a different way.

The first dump hierarchy is used to back up user data (data from user filesets). Because this data changes frequently, it is dumped at the end of each working day, starting with a full dump at the beginning of the week (Sunday) and continuing with incremental dumps Monday through Friday. Each incremental dump refers to the full dump as its parent, rather than to the previous day's incremental dump. As a result, each incremental dump contains all of the data that has changed since the full dump was performed. The following commands are used to establish this dump hierarchy:

```
$ bak adddump /sunday
$ bak adddump /sunday/monday
$ bak adddump /sunday/tuesday
$ bak adddump /sunday/wednesday
$ bak adddump /sunday/thursday
$ bak adddump /sunday/friday
```

You can use the **bak lsdumps** command to display the dump hierarchy. In the following example, **/sunday** is the full dump used for the hierarchy:

$ bak lsdumps

```
/sunday
      /monday
      /tuesday
      /wednesday
      /thursday
      /friday
```

The second dump hierarchy is used to back up filesets containing system binary files. Because these files do not change often, they are backed up only once a week, starting with a full dump at the beginning of the month and followed by an incremental dump at the beginning of each subsequent week. Each weekly dump refers to the previous week's dump as its parent, rather than to the initial full dump. Therefore, each weekly dump contains only those files that changed in the past week, rather than everything that changed since the full dump was performed. The following commands establish this dump hierarchy:

```
$ bak adddump /month
$ bak adddump /month/week1
$ bak adddump /month/week1/week2
$ bak adddump /month/week1/week2/week3
$ bak adddump /month/week1/week2/week3/week4
```

You can use the **bak lsdumps** command to display the dump hierarchy. The following example shows **/month** as the full dump for the hierarchy:

$ bak lsdumps

```
/month
      /week1
            /week2
                  /week3
                        /week4
```

Defining a Dump Level

To define a dump level, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak adddump** command to define one or more dump levels:

   $ bak adddump -level *dump_level*... **[-expires** *date*...**]**

Changing a Dump Level's Expiration Date

To change a dump level's expiration date, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak setexp** command to set the expiration dates of one or more dump levels:

   $ bak setexp -level *dump_level*... **-expires** *date*...

Deleting a Dump Level

To delete a dump level, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak rmdump** command to delete a dump level. All dump levels for which the level serves as the parent, either directly or indirectly, are also deleted automatically.

   $ bak rmdump -level *dump_level*

Labeling Tapes

A tape's magnetic label provides information about the tape and the data it contains. The Backup System checks each tape before it writes to it; if the label on the tape is unacceptable, the dump cannot proceed until you insert an acceptable tape in the drive. A tape's label records the following information:

- The name of the tape, indicating its contents. The name of the tape is composed of three fields, all of which are separated by dots: *fileset_family_name***.** *dump_level***.** *index* (the dump set name with an additional tape index). The following three types of tape names are acceptable:
  - The complete name of the tape in the form *fileset_family_name***.** *dump_level***.** *index*, where the values of *fileset_family_name* and *dump_level* match values that you provide with the **bak dump** command. The *index* is this tape's place in the sequence of tapes used for the complete dump set; if the dump set fits on one tape, the index for that tape is the numeral 1.
  - An indicator of empty, or null, created with the **bak labeltape** command. The Backup System replaces the null indicator with the correct name when it puts dump sets onto the tape.
  - No name, indicating it is an unused tape. Again, the Backup System generates the correct name as it transfers a dump set to the tape.
- The size of the tape. Use a number and a letter (k or K for kilobytes, m or M for megabytes, or g or G for gigabytes) to indicate a size, as described in "Configuring a Tape Coordinator Machine" on page 249 for the **TapeConfig** file. Because the Backup System always uses the size specified in the **TapeConfig** file, the size you include in the label of the tape is intended for information purposes only.

When you label a tape, you can specify its name only, its size only, or both its name and its size. When you dump data to a tape, the expiration date of the dump level at which you dump the data is copied to the label of the tape. If a tape has an expiration date that has not expired, the Backup System refuses to overwrite the tape. If the tape's expiration date has expired, or if the tape contains no expiration date, the Backup System overwrites the tape with a dump set that has an acceptable name.

It is not essential to prelabel tapes before data is transferred to them; the Backup System can use unlabeled tapes or partially labeled tapes, which are tapes that include only the name of the tape or the size of the tape. However, you may want to prelabel a tape in the following situations:

- You want to automate the backup process as much as possible. For manually loaded tape drives, if tapes are prelabeled with the correct name, the individual performing backups needs to respond to prompts only when the system requests a tape. For automated backup devices, if the tapes are prelabeled the backup process will proceed uninterrupted.
- You wish to reuse a tape, putting a different dump set on it. The Backup System will not use a tape if the label reflects the name of a different dump set or if the label contains an unexpired expiration date. Labeling a tape overwrites its expiration date, as well as its name and size.

Note: You can force the Backup System to ignore the tape label and use any unexpired tape for dumps. See "The AUTOQUERY Parameter" on page 256 for information about the **NAME_CHECK** parameter in the user-defined configuration file.

Reading the Label on a Tape

To read the label on a tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See "Starting a Tape Coordinator" on page 247 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. Place the tape in the drive, and issue the **bak readlabel** command to read the label on the tape.

   ```
   $ bak readlabel [-tcid  tc_number]
   ```

Labeling a Tape

To label a tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See "Starting a Tape Coordinator" on page 247 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. Issue the **bak labeltape** command to label the tape. This command executes in the background. (See "Chapter 10. Backing Up and Restoring Data" on page 273 for more information about commands that execute in the background.)

   ```
   $ bak labeltape [-tape tape_name] [-size tape_size] \
   [-tcid  tc_number]
   ```

The **-tape** *tape_name* option is the name of the tape. It must have the form *fileset_family_name***.** *dump_level***.** *index*. If this option is omitted, the tape is marked as empty with the null indicator.

The **-size** *tape_size* option is the capacity of the tape. The default unit is kilobytes. You can add a g or G to the number to indicate gigabytes or an m or M to indicate megabytes. This is for information purposes only; the Backup System uses the tape size recorded in the **TapeConfig** file whenever it uses a tape drive. If this option is omitted, the tape size specified for the drive in the **TapeConfig** file is used.

4. Place the tape in the drive, and press **<Return>** in the corresponding Tape Coordinator's monitoring window.

Adding and Removing Tape Coordinators

As mentioned in "Configuring a Tape Coordinator Machine" on page 249, a separate Tape Coordinator process is associated with each tape drive on a Tape Coordinator machine. Each Tape Coordinator has an associated port, or address. You must assign the port a TCID and use that number in any commands issued to the Tape Coordinator; **bak** commands that involve a tape drive have a **-tcid** option for this purpose. The Backup System identifies and contacts a Tape Coordinator by its TCID.

The Backup System can track a maximum of 1024 tape drives. Valid TCIDs are the integers from 0 to 1023. You do not have to assign the numbers in sequence, and you can skip numbers. The drive with the TCID of 0 is used by default. You can use the **bak lshosts** command to list the Tape Coordinators that have entries in the Backup Database.

"Adding a Tape Coordinator" describes the steps used to add a Tape Coordinator to an existing Tape Coordinator machine. The steps assume that you have already configured the Tape Coordinator machine to which the Tape Coordinator is to be added according to the instructions in "Configuring a Tape Coordinator Machine" on page 249 . You use the **bak addhost** command to add an entry for a Tape Coordinator to the Backup Database.

"Steps Required for a Client-Only Machine" on page 250 describes the steps used to remove an existing Tape Coordinator. You use the **bak rmhost** command to remove an entry for a Tape Coordinator from the Backup Database.

Remember to edit the **TapeConfig** file accordingly when you add or remove a Tape Coordinator. The **TapeConfig** file defines the mapping between a Tape Coordinator and a tape drive on a Tape Coordinator machine.

Adding a Tape Coordinator

To add a Tape Coordinator to an *existing* Tape Coordinator machine, perform the following steps on the Tape Coordinator machine:

1. Install the drive on the machine according to the manufacturer's instructions.
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. Verify that you have the **w** (write) permission on the *dcelocal***/var/dfs/backup/TapeConfig** file.

4. Choose the TCID for the drive. Enter the **bak lshosts** command to check previously assigned TCIDs:

   ```
   $ bak lshosts
   ```

5. Using a text editor, add a line for the new tape drive to the *dceloca*l**/var/dfs/backup/TapeConfig** file. Use a single line in the file for each tape drive, recording the following information:
   - The tape size of the tapes to be used in the drive. The Tape Coordinator uses this capacity as the size of all tapes used in the drive. It is recommended that you use a number 10 to 15% lower than the actual tape capacity to allow for tape variations. The following abbreviations can be used for the tape size unit of measurement (the default is kilobytes); do not leave a space between the number and the letter.
     - Kilobytes: k or K (for example, 2k or 2K)
     - Megabytes: m or M (for example, 2m or 2M)
     - Gigabytes: g or G (for example, 2g or 2G)
   - The EOF mark size for the type of tape to be used in the drive. The Backup System appends an EOF mark after each fileset dumped to tape. The size of this mark can affect the amount of space available for backup data. The EOF mark size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. It is recommended that you increase the actual file mark size by 10 to 15% to allow for tape variations.

     If you do not specify a unit of measurement, the default used for the EOF size is bytes (*not* kilobytes, as for tape capacity). To indicate other units, use the same abbreviations as for tape capacity.
   - The device name of the tape drive. The format of this name varies with each operating system.
   - The TCID for the Tape Coordinator associated with the drive. Legal values are integers from 0 to 1023.

   If you do not know the tape size or EOF mark size for the tape drive, determine them by using the **fms** command described in “Determining Tape Size and End-of-File Mark Size” on page 248.

6. Enter the **bak addhost** command to define an entry in the Backup Database for the Tape Coordinator:

   ```
   $ bak addhost -tapehost machine [-tcid  tc_number]
   ```

Removing a Tape Coordinator

To remove a Tape Coordinator from a Tape Coordinator machine, perform the following steps on the Tape Coordinator machine:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Verify that you have the **w** (write) permission on the *dceloca*l**/var/dfs/backup/TapeConfig** file.
3. Using a text editor, remove the line that defines the Tape Coordinator from the *dceloca*l**/var/dfs/backup/TapeConfig** file.
4. Enter the **bak rmhost** command to delete the entry for the Tape Coordinator from the Backup Database:

   ```
   $ bak rmhost [-tcid tc_number]
   ```

Chapter 10. Backing Up and Restoring Data

Once the DFS Backup System is properly configured, it can be used to help automate the process of making backup copies of both DCE LFS and non-LFS filesets on tape. These copies can then be used to restore data to the file system in the event of data loss. The Backup Database, which stores information about the dump schedule for backups, the locations of the Backup System's Tape Coordinators, the fileset families and their entries that can be dumped, and other administrative information, can also be backed up to tape and restored if the file system becomes damaged or corrupted.

This chapter describes how to use the Backup System to list backup information, back up file system data, and restore data to the file system if necessary. It also details the operations involved in administering the Backup Database, and describes canceling operations from the interactive interface.

The operations in this chapter assume that the Tape Coordinator machines are properly configured (including any special requirements for stackers, jukeboxes, or other automated backup devices), the fileset families and fileset entries are defined, the dump hierarchy is defined, and the required tapes are labeled, as necessary. "Chapter 9. Configuring the Backup System" on page 241 describes configuration of the Backup System in detail; make sure the Backup System is properly configured according to the guidelines detailed in "Chapter 9. Configuring the Backup System" on page 241 before attempting to use it to perform any of the operations described in this chapter.

Introduction to the Backup Process

Backing up, or dumping, data is the most basic operation performed with the Backup System. Data must be dumped to tape before it can be tracked in the Backup Database and before it can be restored from tape to the file system. This section provides an overview of the backup process.

Dumping a fileset makes it inaccessible to other file system users for the duration of the dump process. To reduce inconvenience, create a backup version of a fileset (a version with a **.backup** extension) and dump the backup version rather than the read/write version; this does not interrupt a user's work. Creating a backup version of a fileset, using the **fts clone** or **fts clonesys** command described in detail in "Chapter 6. Making Filesets and Aggregates Available" on page 131, does make its read/write source fileset unavailable for a short period of time; therefore, you may wish to create backup versions during periods of low system usage, using **bos** commands to create a **cron** process to automate the procedure. (See "Chapter 5. Monitoring and Controlling Server Processes" on page 111 for a description of the **bos** commands.)

Occasionally, the Backup System cannot access a fileset, perhaps because of a File Server machine or Fileset Server outage. When this happens, it attempts to access the fileset three times over the course of the operation. If it still cannot access the fileset after the third attempt, it omits the fileset from the dump rather than aborting the dump or waiting for the fileset to become accessible. If the access failure occurs during a full dump, the next incremental dump of the fileset includes the entire fileset; if the failure occurs during an incremental dump, the next incremental dump of the fileset includes all files modified since the last successful

dump of the fileset. You can set the Tape Coordinator performing the dump to notify you of the omission in its monitoring window (by specifying a value of **1** with the **-debuglevel** option of the **butc** command used to start the Tape Coordinator). The Tape Coordinator's error file also records the fileset's omission.

Following is a summary of the process the system uses to perform a typical backup. The example assumes that a backup is being performed on a Wednesday; the fileset family **usersys** is to be dumped at the dump level whose name in the dump hierarchy is **/sunday/wednesday** in this example. Note that the Backup System makes no implied connection between the name of a dump level and the date and time at which a dump at that level is to occur; descriptive dump level names serve merely as reminders to system administrators of when dumps are to be performed.

- The Backup System reads the dump hierarchy in the Backup Database to see if **/wednesday** is an incremental dump and, if so, to determine which preceding level is its parent. In this example, the **/wednesday** level is incremental, and **/sunday** is its parent.

  ```
  /sunday
       /monday
       /tuesday
       /wednesday
       /thursday
       /friday
  ```

  If **/sunday** were specified, this would be a full dump; the system would copy the complete contents of each fileset in **usersys**. Because **/sunday/wednesday** is an incremental dump level, the dump set includes only those files that changed since **usersys** was dumped at the **/sunday** level.
- Because **/sunday** is the parent for **/wednesday**, the Backup System checks the Backup Database for the date and time of the last dump of **usersys** at the **sunday** level.
- The Backup System reads the fileset family **usersys** in the Backup Database to learn which fileset family entries it contains. The fileset family and its entries were created beforehand with the **bak addftfamily** and **bak addftentry** commands. In this example, the entries are

  ```
  .*.* user.*
  .*.* sys.*
  ```
- The Backup System scans the FLDB to match the wildcards from each fileset entry and generates a complete list of the filesets to be included in the dump. If duplicates are found, they are not dumped; only one occurrence of any fileset is included.
- The Backup System reads the label on the tape in the drive to verify that the tape name is acceptable and that the tape does not contain an unexpired expiration date.
- The system transfers the list of filesets to be backed up to the appropriate Fileset Server processes, which determine which data in the filesets was modified after the date and time of the last dump at the **/sunday** level.
- The designated Tape Coordinator puts the gathered data onto tape; the expiration date and other information associated with the dump are stored in the tape's label, and a unique dump ID number is assigned to the dump. If one tape is not large enough to hold the entire dump set, the Backup System prompts the operator to place additional tapes in the drive, as needed.

Standard Information in this Chapter

The following subsections present standard options and arguments common to many of the commands described in this chapter. They also present some common operations repeated throughout this chapter.

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the Reference part of this guide and reference for complete details about each command.)

- The **-family** *fileset_family_name* option is the name of the fileset family to be used in the command. To make it easier to track the contents of a fileset family, its name should give some indication of the contents of the fileset entries it contains (for example, **user** for the fileset family that includes all user filesets in the file system).
- The **-level** *dump_level* option is the name of the dump level to be used in the command. The complete pathname of a dump level must always be specified. There are two types of dump levels:
  - Full dumps, which consist of a name preceded by a single / (slash); for example, **/full**.
  - Incremental dumps, which consist of multiple elements that resemble a pathname listing the dump levels that serve as the parents of the dump level, starting with a full dump level and proceeding in order down the hierarchy; for example, **/full/weekly/monday**.
- The **-tcid** *tc_number* option is the TCID of the Tape Coordinator to be used for the command. Legal values are integers from 0 (zero) to 1023. If this option is omitted, the Tape Coordinator with a TCID of 0 is used to execute the command by default.

Standard Commands and Operations

The following subsections describe commands and operations that are used frequently in this chapter. If a command or operation is described in detail here, it generally is not described in depth in later sections of this chapter where it is used.

Starting a Tape Coordinator

Before performing a backup or restore operation, you must install at least one tape drive on a Tape Coordinator machine and define the tape drive's corresponding Tape Coordinator in both the */dcelocal*/**var/dfs/backup/TapeConfig** file and the Backup Database. If you are using automated backup equipment, you must also create a user-defined configuration file. (See "Chapter 9. Configuring the Backup System" on page 241 for a description of these and other configuration operations that must be performed.)

This section explains how to start a Tape Coordinator. You must have a Tape Coordinator running whenever you access a tape drive for use with the Backup System.

1. Make certain that you have the **w** (write) and **x** (execute) permissions on the */dcelocal*/**var/dfs/backup** directory, which is the directory in which the Tape Coordinator creates its **TL** (log) and **TE** (error) files.

2. Start a new terminal session on the Tape Coordinator machine to use as the monitoring window for the Tape Coordinator. It must remain open while the Tape Coordinator runs.
3. In the newly opened window, issue the **butc** command to start the Tape Coordinator. The binary file for the **butc** program resides in the *dcesharedl***bin** directory.

   $ butc [-tcid *tc_number***] [-debuglevel** *trace_level*]

   The **-debuglevel** *trace_level* option specifies the type of messages to be displayed. There are two valid arguments:

   1 Indicates that the Tape Coordinator is to report on its activities as it restores filesets, in addition to prompting for new tapes as necessary.

   0 Indicates that the Tape Coordinator only prompts for new tapes; it also displays some output as necessary for operations that it executes. This is the default.

Stopping a Tape Coordinator

When you are finished using a Tape Coordinator, you should stop it from running. To stop a Tape Coordinator process, enter an interrupt signal (**<Ctrl-c>** or its equivalent) in the Tape Coordinator's monitoring window.

Using the Interactive Interface

You can use the **bak** commands in regular command mode or in interactive mode. If you use interactive mode, note the following:

- You do not need to enter the string **bak** with each **bak** command; the `bak>` prompt replaces the command shell prompt.
- You do not have to escape regular expression characters; in regular command mode, you must place all regular expression characters in " " (double quotes) or escape each with a **** (backslash).
- You can track executing and pending operations with the **bak jobs** command; in regular command mode, you cannot track operations.
- You can cancel currently executing and pending operations with the **bak kill** command; in regular command mode, you cannot use the **bak kill** command.
- You do not have to establish a new connection each time you issue a command, so execution time is quicker; in regular command mode, each command establishes new connections to the **bakserver** and **flserver** processes, as necessary.

Most of the operations described in this chapter are presented in regular command mode. Where appropriate, some operations include steps introduced as *Optional* to indicate where working in interactive mode could be useful. The **bak jobs** and **bak kill** commands can be entered *only* in interactive mode.

Entering Interactive Mode: Enter the **bak** command to initiate interactive mode:

```
$ bak
```

Leaving Interactive Mode: Enter the **quit** command at the **bak >** prompt to terminate interactive mode:

```
bak> quit
```

Using Commands That Execute in the Background

The following commands used with the Backup System execute in the background:

- **bak dump**
- **bak labeltape**
- **bak restoredb**
- **bak restoredisk**
- **bak restoreft**
- **bak restoreftfamily**
- **bak savedb**
- **bak scantape**

As soon as you enter a command that executes in the background, the prompt at which you entered the command returns to the screen. The command continues to execute, but you can enter additional commands at the prompt while it executes. When you enter a command that does not execute in the background, the prompt does not return until the command is finished executing. (See the appropriate sections in this chapter and in “Chapter 9. Configuring the Backup System” on page 241 for more information about these commands.)

Checking the Status of a Background Operation

You can check the status of a command that is executing in the background by

- Looking in the monitoring window for output from the command
- Entering the **bak status** command
- Entering the **bak jobs** command if you are working in interactive mode

Issue the **bak status** command to check the status of the operation that a Tape Coordinator is currently executing:

```
$ bak status [-tcid number]
```

The command produces output that includes the following:

- A name describing the operation the Tape Coordinator is performing. One or more of the following operation names is displayed:
  - **Dump (** *dump_set*)

    This is displayed for a dump operation initiated with the **bak dump** command; *dump_set* is the name of the dump set in the form *fileset_family_name***.** *dump_level*.
  - **Restore**

    This is displayed for a restore operation initiated with the **bak restoreft**, **bak restoredisk**, or **bak restoreftfamily** command.
  - **Labeltape (** *tape_label*)

    This is displayed for a tape labeling operation started with the **bak labeltape** command; *tape_label* is the label being placed on the tape.
  - `Scantape`

    This is displayed for a tape scanning operation initiated with the **bak scantape** command.
  - `SaveDb`

    This is displayed for a database saving operation initiated with the **bak savedb** command.

– `RestoreDb`

This is displayed for a database restoring operation started with the **bak restoredb** command.

- The number of kilobytes transferred so far (from the file system to tape for a dump operation, or from tape to the file system for a restore operation).
- For a dump operation, the string `fileset` followed by the name of the fileset currently being dumped; for a restore operation, the string `fileset` followed by the name of the fileset currently being restored.
- A message reporting additional status information about the operation. No message is displayed if the operation is proceeding normally.

[abort request rcvd]
The **kill** command was issued, the Tape Coordinator received the **kill**, but the operation is not yet canceled.

[abort complete]
The Tape Coordinator has completed the kill request.

[callout in progress]
The Tape Coordinator is waiting for a callout routine to complete.

[drive wait]
The Tape Coordinator is in the process of locking the device.

[done]
The Tape Coordinator completed the operation.

[operator wait]
The Tape Coordinator is waiting for the operator monitoring the operation to insert a tape in the drive.

The following example shows the status of the operation currently being performed by the Tape Coordinator whose TCID is 0:

```
$ bak status
Dump (usersys.monday):312105 Kbytes transferred,fileset user.terry.
```

(See “Displaying Operations in Interactive Mode” on page 296 for information about the **bak jobs** command.)

Listing Backup Information

The following commands can be used to list information about the Backup Database:

bak verifydb
Checks the status of the Backup Database

bak lsftfamilies
Lists fileset families and fileset family entries

bak lsdumps
Lists the entries in the dump hierarchy

bak dumpinfo
Displays information about specific backups

bak lshosts
Lists Tape Coordinator IDs

bak ftinfo
Displays the dump history for a fileset

In addition to the preceding commands, which display information from the Backup Database, the **bak scantape** command reads a tape, extracting its tape label and information from the fileset header of each fileset on the tape. This command can detect damage to a tape that makes filesets incomplete; if it encounters damage, the scan aborts. All of the commands in the following subsections, like all **bak** commands, require that you be included in the **admin.bak** administrative list.

Verifying Backup Database Status

Issue the **bak verifydb** command to check the status of the Backup Database:

```
$ bak verifydb [-verbose ]
```

The **-verbose** option directs the command to display additional information about the Backup Database.

Following is an example of the output from this command when the database is undamaged:

```
$ bak verifydb
Database OK.
```

Listing Fileset Families and Fileset Family Entries

Issue the **bak lsftfamilies** command to view a fileset family and its entries:

```
$ bak lsftfamilies [-family fileset_family_name]
```

The **-family** *fileset_family_name* option is the name of the fileset family whose entries are to be listed. Omit this option to view all of the fileset families and entries defined in the Backup Database.

Following is an example of the output from this command:

```
$ bak lsftfamilies usersys
Fileset family usersys:
 Entry 1: server.*, aggregate.*, filesets: user.*
 Entry 2: server.*, aggregate.*, filesets: sys.*
```

Listing Entries in the Dump Hierarchy

Issue the **bak lsdumps** command to view the entries in the dump hierarchy:

```
$ bak lsdumps
```

The following example shows a dump hierarchy. The **sunday** entry is the full dump level; the remainder of the entries are incremental dump levels that each have **/sunday** as their parent dump level.

```
$ bak lsdumps
/sunday
      /monday
      /tuesday
      /wednesday
      /thursday
      /friday
```

Viewing Recent Backup Information

Issue the **bak dumpinfo** command to list information about specific backups:

$ bak dumpinfo [{-ndumps *number* | **-id** *dumpID*}] **[-verbose]**

The **-ndumps** *number* option specifies the number of dumps about which information is to be displayed; information about the most recent number of dumps specified with this option is displayed. Use this option or use the **-id** option; omit both options to list information about the last 10 dumps.

The **-id** *dumpID* option specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this option or use the **-ndumps** option; omit both options to list information about the last 10 dumps.

The **-verbose** option includes a detailed list of information about the dump specified with the **-id** option. This option can be used only with the **-id** option.

The dump ID, parent ID, dump level, and other dump information are displayed about the indicated dumps. The following example shows information about the last three dumps:

```
$ bak dumpinfo -ndumps 3
    DumpID   parentID lvl    created     nt nfsets dump_name
-------------------------------------------------------------
 729293644  729289323   1  02/09/93  5:34  1     43 users.tue
 729287531  729286818   1  02/08/93  4:52  1     23 users.mon
 729286056          0   0  02/07/93  4:27  1     31 users.wk1
```

Listing Tape Coordinator TCIDs

Issue the **bak lshosts** command to list all of the Tape Coordinators that have entries in the Backup Database:

```
$ bak lshosts
```

The output lists the name of the machine for which each Tape Coordinator is defined and the TCID of the Tape Coordinator. A Tape Coordinator's presence in the output does not imply that it is currently running.

```
$ bak lshosts

Tape hosts:
Host /.../abc.com/hosts/bak1, port offset 0
Host /.../abc.com/hosts/bak1, port offset 1
Host /.../abc.com/hosts/bak2, port offset 3
Host /.../abc.com/hosts/bak3, port offset 8
Host /.../abc.com/hosts/bak3, port offset 7
Host /.../abc.com/hosts/bak3, port offset 6
```

Displaying a Fileset's Dump History

Issue the **bak ftinfo** command to display the dump history of a fileset:

$ bak ftinfo -fileset *name*

The **-fileset** *name* option names the fileset whose dump history is to be displayed. Include the **.backup** extension if the backup version of the fileset was dumped.

The dump ID, parent ID, dump level, and other dump information are displayed about dumps of the indicated fileset. The following example shows part of the dump history of the *user.smith.backup* fileset:

```
$ bak ftinfo user.smith.backup

   DumpID  parentID lvl creation date   clone date    tape name
654972910  654946323 1  10/01/91 5:07  10/01/91 4:01 users.tuesday.1
654960415  654946323 1  09/30/91  5:11  09/30/91 4:16 users.monday.1
654946323          0 0  09/29/91  5:36  09/28/91 4:31 users.week.1
```

Scanning the Contents of a Dump Tape

To scan the contents of a dump tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
2. Issue the **bak scantape** command to display information from a tape:

   ```
   $ bak scantape [-dbadd] [-tcid tc_number]
   ```

   The **-dbadd** option indicates that information extracted from the tape is to be added to the Backup Database; information is not added if the tape is damaged or if the entry has a dump ID number that is already used by an entry in the Backup Database. (See “Recovering Specific Backup Data” on page 294 for information about using this option.)
3. Place the tape in the drive, and press **Return** in the corresponding Tape Coordinator’s monitoring window. When using this command, you must insert tapes sequentially.

   An example of the output from this command, which lists tape label and fileset information, follows. The output is displayed in the monitoring window of the Tape Coordinator:

   ```
   $ bak scantape

   Tape label
   ----------
   name =         guests.monthly.1
   createTime =    Fri Nov 22 05:59:31 1990
   cell =         /.../abc.com
   size =          20103324 Kbytes
   dump path =     /monthly
   dump id =       729369701
   useCount =      1
   -- End of tape label --



   -- fileset --
   fileset name: user.guest10.backup
   fileset ID 0,,112262
   dumpSetName: guests.monthly
   dumpID 729369701
   level 0
   parentID 0
   endTime 0
   clonedate Fri Nov 22 05:36:29 1991
    . . .
   ```

Backing Up Data

The **bak dump** command is used to perform a dump operation. When you enter the command, specify the fileset family to be dumped and the level that the family is to be dumped. All entries in the specified fileset family are dumped according to the dump level that you specify.

You might perform DFS backups using a DCE ID other than **cell_admin**. In addition to adding this ID to the appropriate groups in the DCE Security Registry and DFS Administration lists, it might be necessary to modify the permissions on the **butc** program so that it runs as a **setuid** program.

If you specify a full dump level, all of the data in all of the filesets included in the specified family is dumped. If you specify an incremental dump, only the fileset data that has changed since the last parent-level dump is dumped.

The fileset family and dump level that you specify produce a dump set. To indicate the contents of the dump set, the dump set name consists of the fileset family name joined by a dot to the last component of the name of the dump level at which the family was dumped. For example, if the fileset family named **usersys** is dumped at the **/weekly/monday** level, the name of the resulting dump set is **usersys.monday**.

The Backup System overwrites a tape that contains an existing dump set only if both of the following conditions are true:

- The tape's label contains an acceptable name. An acceptable name is one that matches the name of the dump set you want to dump to the tape, in the form *fileset_family_name***.** *dump_level***.** *index*. Note that a tape's index is its position in the sequence of tapes necessary to accommodate the dump set; for example, the first tape for a dump set has an index of 1. A tape that is labeled as empty or that has no label is also acceptable.
- The tape's expiration date, if it exists, has expired. The Backup System refuses to overwrite a tape whose expiration date has not expired. Once a tape's expiration date has expired, the Backup System overwrites the contents of the tape with a dump set that has an acceptable name.

Use the **bak labeltape** command to overwrite a label that has an unacceptable name or an unexpired expiration date. Overwriting a tape's label removes all obstacles that can prevent it from being overwritten.

Note: If a dump operation is interrupted or fails for any reason, you cannot be sure that any fileset is complete on one tape; the Backup Database contains an entry for the incomplete dump set, which cannot be used to restore data. Immediately restart the backup, using the same tape to record the dump set; using the same tape automatically removes the entry for the incomplete dump set from the Backup Database. If you use a different tape, you will need to use the **bak deletedump** command to manually remove the entry for the incomplete dump set from the database. (See "Deleting Backup Information" on page 284 for more information about the **bak deletedump** command.)

Using Tapes with a Backup Operation

You must place all dumps of a given fileset family (both full and incremental) onto tapes that are readable by a single tape drive. This is required because a single

Tape Coordinator performs an entire restore operation, using a full dump set and any incremental dump sets as necessary. If a single Tape Coordinator cannot read all of the tapes on which the dump sets are recorded, you cannot restore all of the dumps of the fileset family.

For example, suppose the full dump of a fileset family is stored on 8-mm tape and the incremental dumps, which are done at a different time, are stored on streaming cartridge tape. When you restore a fileset from that fileset family, you must use a Tape Coordinator that uses 8-mm tapes because a restore always begins with the full dump. However, you cannot restore any of the incremental dumps because the same Tape Coordinator cannot read the streaming cartridge tapes and you cannot switch to another Tape Coordinator to continue the restore operation.

Before performing a backup, make sure the tapes are at least as large as the tape size listed in the **TapeConfig** file for the tape drive to be used for the operation. The Backup System fills a tape only with the amount of data listed as the capacity for the drive in the **TapeConfig** file. If a tape is larger than the tape size listed in that file, it simply is not filled to capacity when the backup is performed. However, if the tape is smaller than the size listed in the **TapeConfig** file, the backup operation fails, but only after it fills the tape and determines that it is too small for the drive.

A dump set does not have to fit entirely on a single tape; if the Backup System reaches the end of a tape while dumping a fileset from a fileset family, it puts the remaining data on another tape. The Backup Database automatically records that the fileset resides on multiple tapes.

Prior to performing a backup, you can preview the effects of your command without having the system actually perform the dump. Simply include the **-noaction** option with the **bak dump** command, specifying the remaining options as you would to really execute the dump. This lets you check a fileset family's size before actually dumping it so that you can calculate the correct number of tapes needed.

Backing Up a Fileset (Creating a Dump Set)

To back up a fileset, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See "Starting a Tape Coordinator" on page 275 for information on using the **butc** command to start a Tape Coordinator.)
3. *Optional*. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See "Using the Interactive Interface" on page 276 for the advantages of interactive mode.) The commands in the following steps assume that regular command mode is used, not interactive mode.
4. Decide which fileset family and dump level to use. If necessary, use the **bak lsftfamilies** or **bak lsdumps** command to display information about existing fileset families and dump hierarchies.
5. Check that you have a sufficient number of tapes; if you do not have enough tapes, you will not be able to complete the backup. Also, check that the tapes are *not* smaller than the tape size listed in the **TapeConfig** file for the drive. You must also check that the tapes are properly prelabeled (if necessary, use the **bak readlabel** command to check the labels); you must relabel any tape that
   - Is labeled with an incorrect name; tape names have the following format:

fileset_family_name. *dump_level*. *index*.

- Has an unexpired date; if a tape has an expiration date associated with it from a previous dump, you will not be able to use the tape unless the date is expired.

If a label is incorrect, use the **bak labeltape** command to label the tape correctly.

6. Issue the **bak dump** command to dump the fileset family onto tape:

```
$ bak dump -family fileset_family_name -level   dump_level \
[-tcid tc_number] [-noaction ]
```

The **-noaction** option specifies that all filesets that would be included in the indicated dump be displayed without the dump actually being performed. Specify all other options as you would to actually perform the operation.

7. For manually loaded tape drives, place the correct tape in the drive; the backup process begins immediately. If more than one tape is required, you must remain at the console to respond to prompts for subsequent tapes; if you do not respond immediately, a bell rings periodically to draw your attention. (If you are using automated backup equipment, the user-defined configuration file must be set up to handle tape loading. See “Chapter 9. Configuring the Backup System” on page 241 for details.)

Deleting Backup Information

The Backup System automatically removes the record of a dump set from the Backup Database when the tape containing the dump set is overwritten. The **bak deletedump** command can be used to manually remove information about a dump set from the Backup Database. It can be used to remove the record of a dump set that contains incorrect information (possibly because a dump operation was interrupted or failed) or for which the corresponding tape is to be discarded. Before issuing the **bak deletedump** command, use the **bak dumpinfo** command to display the current dump IDs from the database.

After you use the **bak deletedump** command to delete the record of a dump set from the Backup Database, any dumps for which it serves as the parent, either directly or indirectly, are unusable. You can reissue the **bak deletedump** command to delete those dump sets from the database. However, leaving them in the database, while possibly confusing, causes no problems. Also, as long as the tape that contains the parent dump set remains available, you can always use the **bak scantape** command to restore dump set information about the parent from that tape to the database, making the dump sets that rely on the parent dump set usable again. (See Section 10.6.3 for more information about the **bak scantape** command.)

To delete backup information from the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. *Optional.* At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 276 for the advantages of interactive mode.) The commands in the following steps assume that regular command mode is used, not interactive mode.
3. Issue the **bak dumpinfo** command to list information, including dump IDs, about dump sets recorded in the Backup Database. A dump set’s dump ID is required to delete it from the Backup Database.

```
$ bak dumpinfo [{-ndumps number | -id dumpID}] [-verbose ]
```

The **-ndumps** *number* option specifies the number of dumps about which information is to be displayed; information about the most recent -ndumps is displayed. Use this option or use the **-id** option; omit both options to list information about the last 10 dumps.

The **-id** *dumpID* option specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this option or use the **-ndumps** option; omit both options to list information about the last 10 dumps.

The **-verbose** option includes a detailed list of information about the dump specified with the **-id** option. The **-verbose** option can be used only with the **-id** option.

4. Issue the **bak deletedump** command to delete the desired dump set:

```
$ bak deletedump -id dumpID
```

The **-id** *dumpID* option specifies the dump ID number of the dump set whose entry is to be deleted from the Backup Database.

Restoring Data

The DFS Backup System provides the following commands for restoring data to the file system in different situations:

bak restoreft
: Useful for restoring individual filesets. For example, if a single fileset becomes corrupted or is accidentally deleted, you can use this command to restore the fileset.

bak restoredisk
: Useful for restoring all of the filesets that reside on a single aggregate. For example, if a hardware failure corrupts the partition that houses an aggregate, you can use this command to restore all of the filesets that reside on the aggregate.

bak restoreftfamily
: Useful for restoring all of the filesets that reside on multiple aggregates or multiple File Server machines. For example, if a catastrophic system failure corrupts all of the data on a group of File Server machines, you can use this command to restore all of the filesets that reside on the machines.

The commands that restore data determine the tapes and dumps they need before they begin a restore operation. The commands then prompt for a given tape only once during a restore operation. Before performing a restore operation, you can direct these commands to list the tapes needed to complete the operation. This allows you to locate and assemble the proper tapes before actually issuing the command. To view the list of required tapes, include the **-noaction** option with the command along with any other options you intend to use.

Note: If you have equipment that can automatically retrieve tapes, you can use a user-defined configuration file to override the prompts to mount a particular tape. You can also create executable routines to automatically retrieve the required tapes. See “Chapter 9. Configuring the Backup System” on page 241 for details.

The **-noaction** option of the **bak restoreftfamily** command provides additional information, such as the filesets that were dumped to tape and the sites to which the filesets would be restored. You can use the command's -noaction and **-family** options to generate information about all of the filesets in a fileset family. You can write the information to a file and then modify the file for use with the command's **-file** option. See "Restoring Many Filesets with the bak restoreftfamily Command" on page 290 for information about using the **bak restoreftfamily** command.

Note: If a restore operation is interrupted or fails for any reason, you cannot be sure that any fileset is complete in the file system; immediately restart the operation. If you do not, the file system may contain inconsistent information, which can result in problems in the future.

Specifying the Type and Destination of a Restore Operation

You can perform two types of restore operations: a full restore and a date-specific restore. A full restore recreates a fileset as it existed when it was last dumped, including data from the last full dump and any subsequent incremental dumps. A date-specific restore recreates a fileset as it was when it was last dumped before an indicated date; it includes the last full dump and any incremental dumps done before the indicated date.

The **bak restoredisk** and **bak restoreftfamily** commands always perform full restores of every specified fileset. The **bak restoreft** command performs a full restore of each specified fileset by default, but you can use the command's **-date** option to perform a date-specific restore.

Using any of the three commands, you can restore data to the location that it currently occupies, in which case it overwrites the existing version of the data. Overwrite existing data as follows:

- To use the **bak restoreft** command to restore a fileset to its current location, specify the fileset's existing site with the **-server** and **-aggregate** options.
- To use the **bak restoredisk** command to restore all of the filesets on an aggregate to their current location, omit the **-newserver** and **-newaggregate** options from the command; the filesets are restored to the site at which they currently reside.
- To use the **bak restoreftfamily** command to restore filesets to their current locations, use the **-family** option to specify the fileset families from which filesets are to be restored; the filesets are always restored to their current sites. You can also use the **-file** option of the command to specify a file that names the filesets to be restored and indicates the locations to which they are to be restored; to overwrite the filesets, simply specify their current sites as the locations to which they are to be restored.

Using any of the commands, you can also restore data to a different location, as follows:

- To use the **bak restoreft** command to restore a fileset to a different location, specify a new site with the command's **-server** and **-aggregate** options.
- To use the **bak restoredisk** command to restore all of the filesets on an aggregate to a different location, specify a new site with the **-newserver** option, the **-newaggregate** option, or both.
- To use the **bak restoreftfamily** command to restore filesets to a different location, use the command's **-file** option to specify a file that identifies new sites for the filesets to be restored.

If you plan to specify a new site for a restored fileset, you must remove the current version of the fileset, if it exists, before restoring it. A restore operation fails if a version of a fileset to be restored still exists at a site other than the site to which the fileset is to be restored. If you are using the **bak restoreft** command or the **bak restoreftfamily** command with the **-file** option to restore individual filesets, you can use either the **fts zap** or **fts delete** command to remove the existing filesets. If you are using the **bak restoredisk** command to restore all of the filesets on an aggregate, you can use only the **fts zap** command to remove the existing filesets.

With the **bak restoreft** command, you can preserve the current contents of a fileset in the file system by restoring the data to a new fileset that has the same name as the existing fileset with the addition of a distinguishing extension. You can use the command's **-extension** option to specify an extension such as **.restored** to be appended to the name of the restored fileset. A new FLDB entry is created for the fileset and the fileset is assigned its own fileset ID number. You can restore the new fileset to the same site as the existing fileset or to a different site.

To restore a fileset that no longer exists in the file system and for which an entry no longer exists in the FLDB, use the **bak restoreft** command, or use the **bak restoreftfamily** command with the **-file** option. A new FLDB entry is created for the fileset and a fileset ID number is assigned to it. The **bak restoredisk** command examines the FLDB to determine the filesets to be restored, so you cannot use the command to restore filesets that no longer have entries in the FLDB. The same is true of the **bak restoreftfamily** command when it is issued with the **-family** option.

The following subsections provide detailed information about using the **bak restoreft**, **bak restoredisk**, and **bak restoreftfamily** commands.

Restoring Individual Filesets

Use the **bak restoreft** command to restore one or more individual filesets. Table 10 summarizes the options available with the command. Unless indicated as Optional in the table, each option is required.

Table 10. Options Available with the bak restoreft Command

Option	Specifies	Additional Information
-server	The File Server machine to which to restore each fileset	The specified machine can be a fileset's current site or a different site.
-aggregate	The aggregate to which to restore each fileset	The specified aggregate can be a fileset's current site or a different site.
-fileset	Each fileset to be restored	Specify the name of the read/write version of each fileset to be restored, even if you dumped the `.backup` version of a fileset.
-extension (Optional)	An extension to add to the name of each restored fileset	Specify an extension to preserve filesets in the file system that have the same names as those to be restored. If you want a dot to separate the extension from each name, specify the dot as the first character of the extension (for example, `.restored`).

Table 10. Options Available with the bak restoreft Command (continued)

Option	Specifies	Additional Information
-date (Optional)	The date and, optionally, the time to use for a date-specific restore	Only dump sets of the indicated filesets dated before the specified date are restored. Omit this option to perform a full restore of the most recently dumped version of each fileset. Specify *mm/dd/yy* to indicate 00:00 (12:00 a.m.) on day *mm/dd/yy*; specify "*mm/dd/yy hh*: *mm*" to indicate time *hh*: *mm* on day *mm/dd/yy*. A time must be in 24-hour format (for example, `20:30` for 8:30 p.m.).

To restore individual filesets, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. *Optional.* At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 276 for the advantages of interactive mode.) The command in the following step assumes that regular command mode is used, not interactive mode.
4. Issue the **bak restoreft** command with the appropriate options:

   ```
   $ bak restoreft -server machine -aggregate name -fileset name... \
    [-extension name_extension] [-date date] [-tcid tc_number] \
    [-noaction ]
   ```

   The **-server** *machine* option names the File Server machine to which to restore each fileset. Specify the File Server machine using the machine’s DCE pathname, the machine’s host name, or the machine’s IP address.

   The **-aggregate** *name* option is the device name or aggregate name of the aggregate to which to restore each fileset.

   The **-fileset** *name* option is the name of each fileset to be restored.

   The **-extension** *name_extension* option is a new extension to be added to the name of each fileset when it is restored.

   The **-date** *date* option specifies the date and, optionally, the time to use for a date-specific restore; only dumps performed prior to the specified date (and time) are included in the restore. There are two valid arguments:

   mm/*dd*/*yy*
   : Causes a date-specific restore of dumps that were done before 00:00 (12:00 a.m.) on the indicated date.

   mm/*dd*/*yy hh*:*mm*
   : Causes a date-specific restore of dumps that were done before the specified time on the indicated date. The time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). Surround the entire argument with ″ ″ (double quotes) because it contains a space.

The **-noaction** option directs the command to display the list of tapes needed to complete the restore without performing the actual operation.

Restoring an Aggregate with the bak restoredisk Command

Use the **bak restoredisk** command to restore all of the filesets on an aggregate. Table 11 summarizes the options available with the command. Unless indicated as Optional in the table, each option is required.

Table 11. Options Available with the bak restoredisk Command

Option	Specifies	Additional Information
-server	The File Server machine on which the aggregate that houses the filesets to be restored resides	The filesets on the aggregate are restored to this File Server machine unless the **-newserver** option names a different machine.
-aggregate	The aggregate that houses the filesets to be restored	The filesets on the aggregate are restored to an aggregate of this name unless the **-newaggregate** option names a different aggregate.
-newserver (Optional)	The File Server machine to which the filesets are to be restored	Use this option only to restore the filesets to a File Server machine different from the one specified with the **-server** option.
-newaggregate (Optional)	The aggregate to which the filesets are to be restored	Use this option only to restore the filesets to an aggregate with a name different from the one specified with the **-aggregate** option.

To restore all of the filesets on an aggregate, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. *Optional*. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 276 for the advantages of interactive mode.) The command in the following step assumes that regular command mode is used, not interactive mode.
4. Issue the **bak restoredisk** command with the appropriate options:

   ```
   $ bak restoredisk -server  machine -aggregate name [-tcid tc_number] \
   [-newserver machine] \
   [-newaggregate name] [-noaction ]
   ```

   The **-server** *machine* option names the File Server machine that houses the aggregate to be restored. Specify the File Server machine using the machine’s DCE pathname, the machine’s host name, or the machine’s IP address.

   The **-aggregate** *name* option is the device name or aggregate name of the aggregate to be restored.

   The **-newserver** *machine* option names the File Server machine to which to restore the data. Use this option only if the data is to be restored to a File

Server machine other than the one specified with the **-server** option. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

The **-newaggregate** *name* option is the device name or aggregate name of the aggregate to which to restore the data. Use this option only if the data is to be restored to an aggregate whose name is different from the name of the aggregate specified with the **-aggregate** option.

The **-noaction** option directs the command to display the list of tapes needed to restore the aggregate without performing the actual operation.

Restoring Many Filesets with the bak restoreftfamily Command

Use the **bak restoreftfamily** command to restore filesets that reside on multiple aggregates or on multiple File Server machines. In situations in which you need to restore large amounts of data to multiple sites (for example, when recovering from a catastrophic loss of data), the **bak restoreftfamily** command offers significant advantages over the **bak restoreft** and **bak restoredisk** commands. The **bak restoreftfamily** command provides two ways to restore filesets:

- You can restore all of the filesets that are included in a fileset family to the sites at which they currently reside. (See "Restoring a Fileset Family".)
- You can restore specific individual filesets to the sites at which they currently reside or to different sites, and you can restore different filesets to different sites with a single instance of the command. (See "Restoring Individual Filesets" on page 291 .)

You can use only one of the two approaches with a single instance of the command. Regardless of the method you choose, the command always performs a full restore of all filesets.

Restoring a Fileset Family

To use the **bak restoreftfamily** command to restore all of the filesets included in a fileset family, specify the name of the fileset family with the **-family** option. The command restores the filesets to the sites at which they currently exist.

You can specify the name of an existing fileset family or you can define a new fileset family and add entries that correspond to the filesets you need to restore. The command always reads the FLDB to determine all of the filesets that match the fields of the entries in a specified fileset family, so you can create a fileset family expressly for use with the command. Use the **bak addftfamily** and **bak addftentry** commands to define a new fileset family for use with the **bak restoreftfamily** command. (See "Chapter 9. Configuring the Backup System" on page 241 for information about defining a fileset family.)

In fileset families created for use with the **bak restoreftfamily** command, define entries that match the read/write versions of filesets. The command automatically appends a **.backup** extension to the name of a fileset if it can find no record in the Backup Database of a backup performed for the read/write version. You can include a **.backup** extension to match the backup versions of filesets only if the backup versions were dumped to tape and the backup versions are still valid in the FLDB entries for the filesets.

To restore the filesets included in a fileset family, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. This is optional. (See “Using the Interactive Interface” on page 276 for the advantages of interactive mode.) The command in the following step assumes that regular command mode is used, not interactive mode.
4. Issue the **bak restoreftfamily** command with the **-family** option, as follows:

   ```
   $ bak restoreftfamily -family fileset_family_name [-tcid tc_number] [-noaction ]
   ```

   The **-noaction** option directs the command to display the list of filesets that would be restored and the tapes necessary to restore them without performing the actual operation.

Restoring Individual Filesets

To use the **bak restoreftfamily** command to restore individual filesets, specify the name of a file that includes a single entry for each fileset to be restored. The command restores each fileset to the site specified with the fileset’s entry in the named file. A file to be used with the command must include entries of the following form:

```
machine aggregate fileset [comments...]
```

An entry for a fileset must occupy a single line of the file, and each entry must provide the following information:

machine
: Specifies the File Server machine to which the fileset is to be restored. Identify the machine by its DCE pathname, its host name, or its IP address.

aggregate
: Specifies the aggregate to which the fileset is to be restored. Identify the aggregate by its device name or by its aggregate name.

fileset
: Specifies the fileset to be restored. Specify the name of the read/write version of the fileset. (Note that you can specify the name of the backup version of the fileset if that version was dumped to tape.)

Any *comments* in the form of additional text are always optional; the command treats all remaining text as a comment and ignores it. Do not use wildcards (**.***, for example) in an entry. Note that if a fileset currently exists at a site other than the site you specify, you must remove the existing version of the fileset before issuing the command.

When issued with the **-noaction** option, the **bak restoreftfamily** command generates output that, with slight modification, is suitable for use as input to the command’s **-file** option. You can use the **-noaction** option with the command’s **-family** option to write a list of filesets and their sites to a file. You can then prune the file to include a single entry for each fileset that you need to restore. This approach is useful for restoring only certain filesets from a fileset family or for restoring the filesets to new sites. Note again that the **-family** option provides information only for filesets that have entries in the FLDB.

To restore individual filesets, do the following:

1. Create a file that contains an entry for each fileset you want to restore. Each entry must name the machine to which the fileset is to be restored, the aggregate to which the fileset is to be restored, and the fileset to be restored. Use a single line in the file for each entry, and use a single entry for each fileset (the command ignores subsequent entries for the same fileset). You can use a text editor to create the file manually, or you can use the **-noaction** to write a list of filesets and their sites to a file, which you can then modify for use with the command.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
3. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
4. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. This is optional. (See “Using the Interactive Interface” on page 276 for the advantages of interactive mode.) The command in the following step assumes that regular command mode is used, not interactive mode.
5. Issue the **bak restoreftfamily** command with the **-file** option, as follows:

   ```
   $ bak restoreftfamily -file  filename \
    [-tcid tc_number] [-noaction ]
   ```

   The **-file** *filename* option provides the full pathname of a file that contains an entry for each fileset to be restored.

   The **-noaction** option directs the command to display the list of filesets that would be restored and the tapes necessary to restore them without performing the actual operation.

Administering the Backup Database

A copy of the Backup Database can be installed on any server machine in a cell. The Backup Database stores two types of records used to track all of the backups done in the cell:

- Dump set records, which list the fileset family and the tape used in each dump set
- Administrative records, which list the fileset families, dump levels, and tape hosts

Because information about dumps is difficult to recreate, it is important that you copy the Backup Database with the **bak savedb** command periodically, perhaps weekly. When you issue the **bak savedb** command, the entire database is copied to tape. One tape needs to be designated as a Backup Database tape; when the command is issued, the tape is labeled with the name **bak_db_dump.1**.

If the Backup Database becomes damaged (for instance, if the disk that houses the database becomes damaged), you must delete the old database and restore an entirely new version from its backup tape. You can use the **bak verifydb** command to determine if the Backup Database is damaged.

Do *not* attempt to recover information from a corrupted database. Instead, use the **bos stop** command to shut down *all* **bakserver** processes. Then remove the old Backup Database and its associated files from each machine on which it is located;

the files for the Backup Database are named *dceloca*l**/var/dfs/backup/bkdb.*** on each machine on which the database resides.

Once the database is removed, use **bos start** to restart *all* **bakserver** processes on the machines where they were running. Use **bak addhost** to add a tape host for the Tape Coordinator from which you will restore the Backup Database, and use **bak restoredb** to restore the new version of the database. Recreate fileset information in the database as needed, restoring dump set information that you may have lost since the last backup of filesets; note that any fileset family, fileset family entry, or host information updated since the last backup of the Backup Database must be recreated as well.

If specific information about a dump set is accidentally deleted from the Backup Database, you can use the **bak scantape** command with the **-dbadd** option to check the backup tape used for the dump set, recover the dump set information, and add the information to the Backup Database. Do *not* use the **bak scantape** command to attempt to reconstruct the database.

Backing Up the Backup Database

To back up the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
3. Issue the **bak savedb** command to save the Backup Database to tape:

   ```
   $ bak savedb [-tcid tc_number]
   ```
4. Place the Backup Database tape in the drive, and press **Return** in the corresponding Tape Coordinator’s monitoring window.

Restoring the Backup Database

To restore the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check. In addition to the usual lists, you must also be included in the **admin.bos** list on each machine on which the Backup Database is installed.
2. Verify that you have the **w** (write) and **x** (execute) permissions on the *dceloca*l**/var/dfs/backup** directory on each machine on which the Backup Database is installed.
3. Stop all **bakserver** processes with the **bos stop** command. You must stop all **bakserver** processes on all machines on which the Backup Database is installed.
4. Remove the old Backup Database by deleting the *dceloca*l**/var/dfs/backup/bkdb.*** files from each machine on which the database is installed.
5. Start all **bakserver** processes with the **bos start** command. You must start *all* **bakserver** processes that you stopped in the earlier step; you must restart the processes on the same machines on which they were previously running. When you start a **bakserver** process, an empty Backup Database is created if one does not already exist.

6. Enter the **bak addhost** command to create an entry in the Backup Database for the Tape Coordinator from which you will restore the Backup Database:

   ```
   $ bak addhost -tapehost machine [-tcid tc_number]
   ```

   The **-tapehost** *machine* option is the DCE pathname of the machine (for example, */.../abc.com/hosts/bak1*) for which the Tape Coordinator is to be added.
7. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
8. Issue the **bak restoredb** command to restore the Backup Database to tape:

   ```
   $ bak restoredb [-tcid tc_number]
   ```
9. Place the Backup Database tape in the drive, and press **Return** in the corresponding Tape Coordinator’s monitoring window.

Recovering Specific Backup Data

Use the **bak scantape** command to extract dump set information from a backup tape and add it to the Backup Database. When you issue this command, you must place the backup tapes into the drive in sequential order. The system verifies that each tape is undamaged by checking the end-of-file markers that the Backup System inserts at the beginning and end of each fileset. If the markers are missing, the tape is assumed to be damaged and cannot be used for recovering data. To add information to the database, the entire tape must be undamaged, and the Backup Database must not contain an entry with the same dump ID as an entry being added.

To add recovered data to the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 275 for information on using the **butc** command to start a Tape Coordinator.)
3. Insert the first backup tape from the dump sequence into the tape drive, and issue the **bak scantape** command *without* the **-dbadd** option. Information from the tape is displayed in the Tape Coordinator’s monitoring window.

   ```
   $ bak scantape [-tcid tc_number]
   ```
4. If the output indicates that the tape is undamaged, issue the **bak scantape** command again, *including* the **-dbadd** option. This adds the information from the tape to the Backup Database.

   ```
   $ bak scantape [-dbadd ] [-tcid tc_number]
   ```

   The **-dbadd** option indicates that information extracted from the tape is to be added to the Backup Database; information is added only if the tape is undamaged and the Backup Database does not have an entry with the same dump ID as an entry being added.

Recovering DFS Filesets After DCE Cell Reconfiguration

In the event that you must reconfigure a DCE cell, you can reuse the DCE LFS aggregates and filesets from the original cell. This requires preserving the */dcelocal***/var/dfs/dfstab** file on each DFS File Server machine when the cell is unconfigured (the **rmfilesys.dfs** command leaves this file in */dcelocal***/var/dfs**). When you reconfigure the DFS File Server machine in the new cell, DFS exports the aggregates listed in the **dfstab** file. Then, use the **fts syncfldb** command to update the Fileset Location Database with information about the filesets.

Note: If DCE ACLs exist on DCE LFS files and directories from the previous cell configuration, the ACLs contain incorrect cell UUID information because the cell UUID is changed when the cell is regenerated. To recover from this situation, use the **dcecp acl modify -cell** subcommand to change the default cell of an ACL, giving it the cell name of the local cell that maps to the new UUID. This is necessary even if the cell name has not changed because the cell UUID has changed.

Displaying and Canceling Operations in Interactive Mode

When you issue a command in interactive mode, the resulting operation is assigned a unique job ID number. While in interactive mode, you can use the **bak jobs** command to list job ID numbers and status information about all of the operations currently executing or pending in the queue on a tape drive (operations that do not involve tapes execute immediately and do not appear on the list). You can use the job ID number of an operation (or its dump set name if it is a dump) with the **bak kill** command to cancel an operation that is executing or that is in the queue.

If you cancel an operation that is in the queue, it is removed from the queue with no other effects. Furthermore, if you cancel a tape labeling or tape scanning operation as it executes, the operation simply terminates with no further effects. However, canceling a dump or restore operation while it executes can produce inconsistencies on a backup tape or in the file system.

If you cancel a backup operation while it is executing, all filesets written to tape *before* the **kill** signal is received are complete and usable on the tape. However, filesets being written *when* the signal is received may be incomplete and *should not be used*.

If you cancel a restore operation while it is executing, all completely restored filesets are online and usable. However, because most restore operations require data from multiple tapes (a full dump tape and one or more incremental dump tapes), most filesets are usually not completely restored. If the **kill** signal occurs before the system accesses all of the necessary tapes, most filesets are not restored to the desired date or version and *should not be used*.

If the interrupted restore operation is overwriting one or more existing filesets, the filesets can be lost entirely; however, the data being restored still exists on tape. In general, to avoid the inconsistencies that can result from an interrupted restore operation, reinitiate the restore operation.

Displaying Operations in Interactive Mode

Issue the **bak jobs** command to determine the job ID number of an operation. For an operation to appear in the output from the **bak jobs** command, you must have initiated the operation in interactive mode, and you must still be in interactive mode. No privileges are required to display an operation with the **bak jobs** command. (See “Using the Interactive Interface” on page 276 for more information about interactive mode).

```
bak> jobs
```

If no operations are pending or executing, the prompt returns immediately. Otherwise, the output reports the following information for each job. The output is very similar to that produced by the **bak status** command.

- The job ID number.
- A name describing the operation. One of the following operation names is displayed for each job:
  - `Dump` (*dump_set*)

    This is displayed for a backup operation initiated with the **bak dump** command; *dump_set* is the name of the dump set in the form *fileset_family_name***.** *dump_level*.
  - `Restore`

    This is displayed for a restore operation initiated with the **bak restoreft**, **bak restoredisk**, or **bak restoreftfamily** command.
  - `Labeltape` (*tape_label*)

    This is displayed for a tape labeling operation started with the **bak labeltape** command; *tape_label* is the tape label specified with the **bak labeltape** command’s options.
  - `Scantape`

    This is displayed for a tape scanning operation initiated with the **bak scantape** or **bak readlabel** command.
  - `SaveDb`

    This is displayed for a **bak savedb** operation.
  - `RestoreDb`

    This is displayed for a **bak restoredb** operation.
- The number of kilobytes transferred so far (from the file system to tape for a dump operation, or from tape to the file system for a restore operation).
- For a dump operation, the string `fileset` followed by the name of the fileset currently being dumped; for a restore operation, the string `fileset` followed by the name of the fileset currently being restored.
- A message indicating the status of the operation. No message is displayed if the operation is executing normally.

  [abort request]
  : The **kill** command was issued, but the operation is not yet canceled.

  [abort sent]
  : The operation is canceled. Once the system removes it from the queue or stops its execution, the operation no longer appears in the listing from the **bak jobs** command.

  [callout in progress]
  : The Tape Coordinator is waiting for a callout routine to complete.

butc contact lost
The **bak** program temporarily lost contact with the Tape Coordinator executing the program.

[drive wait]
The operation is waiting for the specified tape drive to become free.

[operator wait]
The Tape Coordinator is waiting for the operator monitoring the operation to insert a tape in the drive.

Canceling Operations in Interactive Mode

Issue the **bak kill** command to cancel an operation. Use the **bak jobs** command to determine the job ID number of the operation to be killed. No privileges are required to cancel an operation with the **bak kill** command. The command can be issued only in interactive mode. (See “Using the Interactive Interface” on page 276 for more information about interactive mode.

bak> kill {*jobID* | *dump_set*}

The *jobID* argument is the unique job ID number of the operation to be canceled; the *dump_set* argument is the name of the operation in the form *fileset_family_name.dump_level* if it is a dump. Use either argument to indicate the operation to be canceled.

Chapter 11. Monitoring and Tracing Tools

DFS provides two tools to help you monitor and probe the file system:

- The **scout** program is used to monitor the File Exporter process on a File Server machine.
- The **dfstrace** command suite is used to diagnose problems associated with the DFS kernel or with DFS server processes.

Monitoring File Exporters with the scout Program

The **scout** program can help you monitor the File Exporter running on any File Server machine. When you use **scout**, it periodically collects statistics from the File Server machines that you specify and displays the information in a graphical format. The program has several useful features:

- You can monitor the File Exporters on several File Server machines, including the local machine and machines in other cells, from one location.
- You can set attention thresholds for many of the statistics. When a value exceeds the specified threshold, **scout** highlights it with reverse video; when the value drops below the threshold, **scout** removes the highlighting.
- If the File Exporter on a machine does not respond to **scout**'s probes, the program highlights the machine's name on the screen and blanks the other fields for the machine. When the File Exporter returns to service, the machine name and the statistics are again displayed normally.

An Overview of the scout Program

You can run **scout** from any machine configured as a DFS client or server. In a standard configuration, the binary file for **scout** is stored in the *dceshared*/**bin/scout** file. Both terminals and windowing systems that emulate terminals can display **scout**'s statistics; the display appears best on systems that support reverse video and cursor addressing.

When using **scout**, set the **TERM** environment variable (or its equivalent) to the correct terminal type or to one with characteristics similar to the actual ones. Do not resize the window while **scout** is running; the program does not adjust to the new dimensions if the window is made larger, and, if it is made smaller, the columns may not align properly. To resize the window, stop **scout**, resize the window, and then restart the program.

Although no special privileges are required to run **scout**, it is useful primarily for system administrators who need to monitor File Exporter usage. While **scout** imposes only a minimal burden on the File Exporter on a File Server machine, you may wish to place the binary file for the program in a secure directory that is available only to administrative users. Other users are then prevented from needlessly running the program.

You can run multiple **scout** processes from a single machine; because the **scout** program must run in the foreground, each instance runs in a separate, dedicated window. You can also run **scout** on several machines and view the output on a single machine. To do this, open several windows on one machine, log into different remote machines in each window (by using **telnet** or an appropriate program), and

run **scout** in the windows. You may find it useful to include the **-host** option with the **scout** command; this option marks each window with the name of the host machine that is running the program.

You use the **-server** option of the **scout** command to indicate each File Server machine whose File Exporter is to be monitored. You can specify machines by DCE pathname, hostname, or IP address.

In most cells, the DCE pathnames of all File Server machines share the same basename, or common DCE prefix; for example, all File Server machines in the cell *abc.com* have DCE pathnames of the form */.../abc.com/hosts/ hostname*. If all of the machines to be monitored have the same DCE pathname prefix, you can specify the common prefix (for example, */.../abc.com/hosts*) with the **-basename** option and specify only the unique part of each machine name (for example, **fs1** or **fs2**) with the **-server** option. If you use the **-basename** option, the common prefix you specify is displayed on the screen (see "The scout Screen").

The scout Screen

The output generated by the **scout** program appears in the following three main regions on the screen:

- The banner line at the top of the screen displays the word `scout`, indicating that the program is running; it can display additional information if you include the following options:
  - The **-host** option displays the name of the machine executing the **scout** program.
  - The **-basename** option, if specified, displays the common DCE pathname prefix of all File Server machines being monitored.
- The statistics display region comprises the majority of the window. In this region, **scout** displays the statistics gathered for each File Exporter; it displays each File Exporter on its own line. The area is divided into six columns, with the following labels and information:
  - `Conn`

    The number of open connections with principals. This column displays the number of connections that principals (users and machines) have open to the File Exporter. This number usually exceeds the number in the `Ws` column, which shows the number of active client machines, because each user on a machine can have several separate connections open at once and because one client machine can handle several users.
  - `Fetch`

    The number of fetches sent from client machines to the File Exporter. A fetch is an RPC requesting that the File Exporter send data. The statistic represents the number of fetches since the File Exporter started; it is reset to 0 (zero) whenever the File Exporter restarts.
  - `Store`

    The number of stores sent from client machines to the File Exporter. A store is an RPC requesting that the File Exporter store data. The statistic represents the number of stores since the File Exporter started; it is reset to 0 (zero) whenever the File Exporter restarts.
  - `Ws`

    The number of active client machines. This column shows the number of client machines that have communicated with the File Exporter in the last 15

minutes. This number is usually smaller than the value for `Conn` because a single client machine can have several connections open to one server.

– `Server`

The File Server machine name. This column contains the name of the File Server machine where the File Exporter is running. Names are shortened to display only the first 10 characters of the unique part of each machine name, which are the characters that follow the **hosts** element of a DCE pathname. If two or more machines from different cells have a common name (for example, */.../abc.com/hosts/fs1* and */.../def.com/hosts/fs1*), the name of each cell followed by a colon is displayed before the name of each machine (*abc.com:fs1* and **def.com:fs1**). If the name of a File Server machine is too long for the width of the column, an ***** (asterisk) can appear in place of one or more characters in the name.

– `Disk attn`

The disk usage. This column displays the number of kilobyte blocks available on each DFS aggregate on the File Server machine. For example, a value of `/dev/lv01:8949` indicates that the aggregate **/dev/lv01** has 8949 kilobyte blocks free. If the window is not wide enough for all of the aggregate names, **scout** automatically creates subcolumns for the information. As with the names of server machines, if the name of an aggregate is too long for the width of the column, an ***** (asterisk) can appear in place of one or more characters in the name.

The label on this column indicates the threshold value at which entries become highlighted. By default, **scout** highlights the entry for any aggregate that is over 95% full. Therefore, the default label for this column appears as `Disk attn:> 95% used`.

For all columns except the `Server` column, you can use the **-attention** option to set a threshold at which entries in the column are highlighted. This notifies you that a certain value is exceeded. `Disk attn` is the only statistic with a preset default.

- The message/probe line at the bottom of the screen indicates how many times **scout** probed the File Exporters for statistics. By default, **scout** probes every 60 seconds; you can use the **-frequency** option to specify a different rate of time.

Setting Attention Thresholds

The **-attention** option can be used to set the threshold value for all but the `Server`

column in the display region. Any threshold that you set applies to all of the entries in a column; you cannot set thresholds on a per-machine basis.

You can use more than one argument with the **-attention** option; each argument is a combination of a statistic and a threshold. Legal values for statistic/threshold pairs are as follows:

- The **conn** *connections* argument sets the threshold for the maximum number of connections that principals can have open to the File Exporter before the value is highlighted. The highlighting is removed when the value goes below the threshold.
- The **fetch** *number_of_fetches* argument sets the threshold for the maximum number of fetches the File Exporter can service before the value is highlighted. The highlighting is removed when the File Exporter is restarted.

- The **store** *number_of_stores* argument sets the threshold for the maximum number of stores the File Exporter can accept before the value is highlighted. The highlighting is removed when the File Exporter is restarted.
- The **ws** *active_client_machines* argument sets the threshold for the maximum number of active client machines that the File Exporter can serve before the value is highlighted; **active** indicates those machines that communicated with the File Exporter in the past 15 minutes. The highlighting is removed when the value goes below the threshold.
- The **disk** *percent_full***%** argument sets the threshold for the maximum percentage of an aggregate that can contain data before the value is highlighted. This threshold is applied to each exported aggregate or partition on a File Server machine being monitored. Legal thresholds are integers from 0 (zero) to 99; the default is `95% used`. You *must* enter the % (percent sign) with this threshold; if the % (percent sign) is absent, **scout** interprets the number as a number of kilobyte blocks.

  or

  The **disk** *minimum_blocks_free* argument sets the threshold that determines the minimum number of kilobyte blocks to be available on the aggregate before the value is highlighted. This threshold is applied to each exported aggregate or partition on a File Server machine being monitored.

To change these attention settings, you must stop and restart **scout**. In addition, **scout** does not store the settings from previous instances; you must specify the desired settings each time you start the program.

You cannot set a threshold or control the highlighting in the `Server` column. If the File Exporter on a machine does not respond to **scout**'s probes, the name is automatically highlighted and the values for that machine in the other columns are removed. A lack of response can indicate that a File Server machine or the network is unavailable.

When a machine resumes responding to **scout**'s probes, its name is displayed without highlighting with the other values on its line of the display. If all of the machine names are highlighted simultaneously, a network outage has possibly disrupted the connections between the File Server machines and the client machine running **scout**.

The following examples demonstrate the different types of **-attention** settings. The first example causes **scout** to highlight entries in the `Conn` column that exceed 100, entries in the `Ws` column that exceed 20, and entries in the `Disk attn` column that reflect aggregate fullness usage of 75% or more on the machines named */.../abc.com/hosts/fs1* and */.../abc.com/hosts/fs2*:

```
$ scout -server /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 \
-attention conn 100 ws 20 disk 75%
```

The second example is identical to the previous example, except that **scout** highlights entries in the `Disk attn` column that fall below 5000 free blocks:

```
$ scout -server  /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 \
-attention disk 5000 ws 20 conn 100
```

The third example causes **scout** to highlight entries in the `Fetch` column that equal or exceed 20,000 fetches:

```
$ scout -server  /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 \
-attention fetch 20000
```

Using the scout Program

Start the **scout** program by issuing the **scout** command to initialize it in each window in which you want it to run. No further commands can be issued in the window as long as the program is running. To stop **scout**, enter the interrupt command (**<Ctrl-c>** or its equivalent) for your system in the window in which **scout** is running.

Starting the scout Program

To start the **scout** program, do the following:

1. Open a command shell window for each instance of **scout** you want to run.
2. Initialize **scout** in each window; note that you will not be able to issue any further commands in the window as long as **scout** is running:

   ```
   $ scout -server machine... [-basename  common_prefix] [-host ] \
    [-frequency  seconds] [-attention  stat/threshold_pair...] \
    [-debug filename]
   ```

   The **-server** *machine* option specifies each File Server machine whose File Exporter is to be monitored. Use one of the following to indicate each File Server machine:

   - The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*). If you use the **-basename** option to specify a pathname prefix common to all machines to be monitored, you need to provide only the unique suffix of each machine name; you can omit the common DCE pathname prefix.
   - The machine's hostname (for example, **fs1.abc.com** or **fs1**).
   - The machine's IP address (for example, **11.22.33.44**).

   The **-basename** *common_prefix* option specifies the DCE pathname prefix (for example, */.../abc.com/hosts*) common to all File Server machines specified with the **-server** option. Do not include the **/** (slash) that separates the prefix from the unique part of each machine name; it is included automatically with the **-basename** option. The basename, if specified, is displayed in the banner line.

   Use this option only if you are specifying the DCE pathname of each File Server machine to be monitored. Omit this option if you are specifying the hostnames or IP addresses of one or more machines.

   The **-host** option displays the name of the machine running the **scout** program in the banner line; this is useful if you are logged into the machine remotely.

   The **-frequency** *seconds* option indicates how often the **scout** program is to probe the File Exporters. Specify a positive integer as a value in seconds; the default is 60 seconds.

   The **-attention** *stat/threshold_pair* option specifies a list of attention settings (statistic and threshold pairs); **scout** highlights any value for a statistic that exceeds its threshold. (See "Setting Attention Thresholds" on page 301 for a discussion of legal values for this argument.)

   The **-debug** *filename* option enables debugging output and directs it to the specified *filename*. Provide a complete pathname for *filename*; the current working directory is used by default. If this option is omitted, no debugging output is written.

The following example causes **scout** to monitor the File Exporters on the File Server machines named **fs1** and **fs2** in the cell named *abc.com*; the **-basename** option is used, so the common DCE prefix is specified only once, and it appears in the banner line. The **scout** program probes the File Exporters every 30 seconds and prints debugging information to the file named */.../abc.com/fs/usr/terry/scout.one*.

```
$ scout -server fs1 fs2 -basename /.../abc.com/hosts -frequency 30 \
 -debug /.../abc.com/fs/usr/terry/scout.one
```

The following example again instructs **scout** to monitor the File Exporters on the machines **fs1** and **fs2** in the cell *abc.com*; the **-basename** option is again used to indicate the common prefix. The **-host** option is used, so the name of the machine running **scout** appears in the banner line with the basename prefix. The **scout** program highlights an entry in the `Fetch` column if 30,000 or more fetches are sent to the File Exporter; the program highlights an entry in the `Store` column if 20,000 or more stores are sent to the File Exporter.

```
$ scout -server  fs1 fs2 -b /.../abc.com/hosts -host -attention fetch 30000 \
 store 20000
```

Stopping the scout Program

To stop the **scout** program, enter the interrupt command (**<Ctrl-c>** or its equivalent) for your operating system in the **scout** window.

Tracing DFS Kernel and Server Process Events with the dfstrace Command Suite

To properly diagnose a DFS problem, it is sometimes necessary to analyze in detail the workings of the DFS kernel and DFS server processes. The **dfstrace** command suite provides commands that allow you to perform this analysis and related tasks.

An Overview of the dfstrace Command Suite

The **dfstrace** command suite allows a system administrator to manipulate the two main elements of the underlying DFS trace facility: the event set and the trace log.

An *event set* is a logical grouping of related kernel or server process events. Events with similar characteristics are grouped into event sets to aid in the diagnosis of specific types of problems. When an event in an event set occurs, a message is logged to each appropriate trace log; event set information from one event set can be written to from one to eight trace logs.

An event set can be in one of three states:

- `active` − Tracing is enabled for the event set.
- `inactive` − Tracing is temporarily disabled for the event set; however, the event set continues to claim space occupied by the logs to which it sends data.
- `dormant` − Tracing is disabled for the event set; furthermore, the event set releases its claim to space occupied by the logs to which it sends data. When all of the event sets that send data to a particular log are in this state, the space for that log is deallocated.

Following are some of the DFS kernel event sets that you can trace:

- **cm** − Cache Manager package
- **fshost** − File exporter host package

- **fx** − File exporter package
- **episode/anode** − DCE LFS anode package
- **episode/logbuf** − DCE LFS buffer/logging package
- **episode/vnops** − DCE LFS vnode package
- **tkc** − Token cache package
- **tkm** − Token manager package
- **tpq** − Thread pool queue package
- **xops** − Vnode-to-fileset synchronization package

Following are some of the server process event sets that you can trace:

- **bosserver** − **bosserver** package
- **dacl** − DFS ACL package
- **dfsauth** − DFS security package
- **flserver** − **flserver** package
- **ftserver** − **ftserver** package
- **ftutil** − Fileset utility package
- **ubikdisk** − Disk I/O subset of Ubik package
- **ubikvote** − Sync site election subset of Ubik package

A *trace log* is a piece of memory within the kernel or server process where event information is written when an event associated with an `active` event set occurs. Multiple event sets that contain related events often write information to the same log to aid in problem diagnosis. Every trace log has a default size assigned to it; however, the size of a log can be increased or decreased.

A trace log can either be `allocated` or `deallocated`:

- When a trace log is `allocated`, space is allocated for it in the kernel or server process memory. A trace log is `allocated` when one or more of the event sets that write to the log are either active or inactive.
- When a trace log is `deallocated`, no space is allocated for the log in the kernel or server process memory. A trace log is `deallocated` when all of the event sets that write to the log are dormant.

Both event sets and trace logs can have a persistence attribute permanently assigned to them. If an event set is `persistent`, its state cannot be set as part of a global event set state setting (performed with the **dfstrace setset** command), but its state can be manipulated if indicated explicitly. If a trace log is `persistent`, it cannot be cleared as part of a global log clearing (performed with the **dfstrace clear** command), but it can be cleared if indicated explicitly.

The dfstrace Commands

The **dfstrace** commands allow you to manipulate the event sets and trace logs that make up the underlying DFS trace facility. You can perform the following functions with the **dfstrace** commands:

- List information about event sets with the **dfstrace lsset** command. You can list event sets and get state and persistence information on each set.
- Set an event set's state with the **dfstrace setset** command. You can set an event set's state to `active`, `inactive`, or `dormant`.
- List information about trace logs with the **dfstrace lslog** command. You can list trace logs and get size and allocation information on each log.

- Change the size of trace logs with the **dfstrace setlog** command. The size of trace logs are specified in 4-kilobyte units (kwords).
- Dump the contents of trace logs with the **dfstrace dump** command. You can dump kernel trace logs to standard output or to a file; you can also continuously dump a kernel trace log. You can dump server process logs to a file only.
- Clear trace logs with the **dfstrace clear** command. You can completely remove the current contents of one or more trace logs.

General Recommendations for Diagnosing DFS Problems

Use the following general guidelines when diagnosing a DFS problem:

1. When a problem occurs, determine the event sets related to the problem and set their states to `active` using the **dfstrace setset** command. Messages from multiple event sets are indistinguishable from each other if they are written to a single log; therefore, by disabling event sets whose events are unrelated to a problem, you can more easily diagnose that problem.
2. If a problem is reproducible, clear the appropriate logs with the **dfstrace clear** command and reproduce the problem. If the problem is not easily reproduced, keep tracing enabled until the problem recurs.
3. Dump the appropriate logs using the **dfstrace dump** command.

Note: When you are not diagnosing problems, set all event sets to the `dormant` state using the **dfstrace setset** command. When tracing is enabled, system performance is affected.

Standard Information on the dfstrace Command Suite

This section presents options and arguments common to many of the **dfstrace** commands described in this chapter. It also presents a common operation that can be useful when performing **dfstrace** operations.

Standard Options and Arguments

The following options and arguments are used with many of the **dfstrace** commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See Part 2 of this guide and reference for complete details about each command.)

- The **-log** *log_name* option specifies the name of a server process or kernel trace log to be utilized by the command. All logs are stored in kernel or server process memory that is allocated on the initialization of the kernel or server process. The default size of a log, which is measured in kwords, is predefined; however, this size can be changed. Any number of event sets can write to a single log.
- The **-cdsentry** *server_entry_in_CDS* option specifies the name of a server process to which to connect. This option is required when performing operations on server process logs and event sets; it must be omitted when performing operations on kernel logs and event sets. The full DCE pathname of a server process must be specified with this option (**/.:/hosts/***machine*/*process_name*).

Determining Administrative Privilege

To perform the majority of **dfstrace** commands on a server process event set or log, the issuer must be listed in the appropriate administrative list (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process). To determine the members of a list, issue the **bos lsadmin** command:

```
$ bos lsadmin -server machine -adminlist filename
```

The **-server** *machine* option names the server machine that houses the administrative list whose principals and groups are to be displayed. The BOS Server on this machine executes the command. Specify the File Server machine using the machine's DCE pathname, the machine's hostname, or the machine's IP address.

The **-adminlist** *filename* option specifies that members of the administrative list *filename* to be listed.

Note: To perform **dfstrace** commands on a kernel event set or log, the issuer must be logged into the local machine as **root**.

Listing Information about Event Sets

The **dfstrace lsset** command provides state and persistence information on specified event sets. To list information about event sets, do the following:

1. Verify that you have the necessary privilege. To list information about a kernel event set, you must be logged in as **root** on the local machine. To list information about a server process event set, you must be listed in the administrative list associated with that process on the corresponding machine; if necessary, issue the **bos lsadmin** command to check.
2. Issue the **dfstrace lsset** command to list information about each specified event set:

   ```
   $ dfstrace lsset [-set set_name...] \
   [-cdsentry server_entry_in_CDS]
   ```

   The **-set** *set_name* option specifies the name of each event set you want to list. Omit this option to list all kernel event sets on the local machine or all server process event sets for the server process specified with the **-cdsentry** option.

   The command lists each event set and its state; if the event set is persistent, the word `persistent` appears after the state. If an event set is `persistent`, its state cannot be set during a global state setting (executed by issuing the **dfstrace setset** command with the **-set** option). Of course, the event set's state can still be set if the event set is otherwise specified with the **dfstrace setset** command. The `persistent` attribute prevents accidental resetting of an event set's state. The attribute is assigned to an event set when the kernel or server process is compiled and it cannot be changed.

The following command lists all kernel event sets and their states on the local machine:

```
# dfstrace lss
Available sets:
cm: active
fx: active
fshost: active
xops: active
episode/anode: dormant
episode/logbuf: dormant
episode/vnops: dormant
tkc: inactive
tpq: active
tkm: active
```

The following command lists state information on the event set **ubikvote** for the **flserver** process on the machine named **fs1**:

```
$ dfstrace lss -set ubikvote -cdsentry /.:/hosts/fs1/flserver
ubikvote: active
```

Setting an Event Set's State

An event set's state determines whether information on the events in that event set is logged to the event set's trace logs. To set an event set's state, do the following:

1. Verify that you have the necessary privilege. To set the state of a kernel event set, you must be logged in as **root** on the local machine. To set the state of a server process event set, you must be listed in the administrative list associated with that process on the corresponding machine; if necessary, issue the **bos lsadmin** command to check.
2. Issue the **dfstrace setset** command to set the state of each specified event set:

   ```
   $ dfstrace setset [-set  set_name...] [{-active  | \
    -inactive  | -dormant }] [-cdsentry server_entry_in_CDS]
   ```

   The **-set** *set_name* option specifies the name of each event set whose state you want to set. Omit this option to set the state for all non `persistent` kernel event sets on the local machine or all non `persistent` server process event sets for the server process specified with the **-cdsentry** option.

   The **-active** option sets the state of each specified event set to `active`. This option enables tracing for each specified event set. Use this option or the **-inactive** or **-dormant** option.

   The **-inactive** option sets the state of each specified event set to `inactive`. This option disables tracing for each specified event set; however, each event set continues to claim space occupied by each log to which it sends data. Use this option or the **-active** or **-dormant** option.

   The **-dormant** option sets the state of each specified event set to `dormant`. This option disables tracing for each event set; furthermore, each event set releases its claim to space occupied by each log to which it sends data. Use this option or the **-active** or **-inactive** option.

The following command sets the event state of all non- `persistent` kernel event sets on the local machine to `inactive`:

```
# dfstrace –inactive
```

Listing Information about Trace Logs

The **dfstrace lslog** command provides size and allocation information on specified trace logs. To list information about a trace log, do the following:

1. Verify that you have the necessary privilege. To list information about a kernel log, you must be logged in as **root** on the local machine. To list information about a server process log, you must be listed in the administrative list associated with that process on the corresponding machine; if necessary, issue the **bos lsadmin** command to check.
2. Issue the **dfstrace lslog** command to list information about each specified log:

   ```
   $ dfstrace lslog [{-set set_name... | -log log_name...}] \
   [-long ] [-cdsentry server_entry_in_CDS]
   ```

The **-set** *set_name* option specifies the name of each event set whose corresponding logs you want to list. Specify this option or the -log option; omit both to list all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option.

The **-long** option directs the **dfstrace lslog** command to also provide information on the size of each log in kwords and whether the log is physically allocated in the kernel or server process memory.

When run *without* the **-long** option, the **dfstrace lslog** command lists the logs only. When run with the **-long** option, the command lists the logs, the size of each log in kwords, and the allocation state of each log.

A log can also be `persistent`; however, the persistence of a log cannot currently be determined using **dfstrace** commands. If a log is `persistent`, it cannot be cleared during a global log clearing (executed by issuing the **dfstrace clear** command without the **-set** or **-log** option). Of course, the log can still be cleared if it is otherwise named with the **dfstrace clear** command. The `persistent` attribute prevents accidental clearing of important logs. The attribute is assigned to a log when the kernel or server process is compiled and cannot be changed.

The following command lists all kernel logs on the local machine:

```
# dfstrace lsl
Available logs:
cmfx
DFS syslog
```

The following command lists all server process logs used by the **flserver** process on the machine named **fs1**; it also provides the size and the allocation status of each log.

```
$ dfstrace lsl -long -cdsentry /.:/hosts/fs1/flserver
Available logs:
ubikvote : 30 kwords (allocated)
common : 30 kwords (allocated)
```

Changing the Size of Trace Logs

By default, trace logs occupy a specific amount of kernel or server process memory; however, the size of this memory space can be changed. To change the size of a trace log, do the following:

1. Verify that you have the necessary privilege. To change the size of a kernel log, you must be logged in as **root** on the local machine. To change the size of a server process log, you must be listed in the administrative list associated with that process on the corresponding machine; if necessary, issue the **bos lsadmin** command to check.
2. Issue the **dfstrace setlog** command to set the state of each specified event set:

   $ dfstrace setlog -log *log_name* **-buffersize** *4_kilobyte_units* ****
   [-cdsentry *server_entry_in_CDS*]

   The **-buffersize** option defines the size of the log in kwords. If a specified log is already allocated, it is cleared and freed when this command is run, and a new log of the desired size is created. Otherwise, a log of the desired size is created when the log is allocated.

The following command sets the size of the **ubikvote** server process log on the machine named **fs1** to 120 kilobytes (30 kwords):

```
$ dfstrace setl ubikvote 30 -cdsentry /.:/hosts/fs1/flserver
```

Dumping the Contents of Trace Logs

To view the information contained in a trace log, you must dump the log. You can dump a kernel log to standard output or to a file; you can also continuously dump a kernel log. You can dump a server process log to a file only. To dump a trace log, do the following:

1. Verify that you have the necessary privilege. To dump a kernel log, you must be logged in as **root** on the local machine. To dump a server process log, you must be listed in the administrative list associated with that process on the corresponding machine; if necessary, issue the **bos lsadmin** command to check.
2. Issue the **dfstrace dump** command to dump each specified log:

   ```
   $ dfstrace dump [{-set  set_name... | -follow log_name}] \
    [-file output_filename] [-sleep seconds_between_reads] \
   [-cdsentry server_entry_in_CDS]
   ```

   The **-set** *set_name* option specifies the name of each event set whose corresponding logs you want to dump. Specify this option or the -follow option; omit both to dump all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option. If you specify multiple event sets that point to the same log, that log is dumped multiple times.

   The **-follow** *log_name* option specifies the name of a kernel log to continuously dump. Process server logs cannot be continuously dumped. When a log is continuously dumped, it is also cleared. Specify this option or the **-set** option; omit both to dump all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option.

   The **-file** *output_filename* option indicates the name of a file to which to write the output of the command. If the log being dumped is a server process log, the *output_filename* cannot contain slashes (/); the file is automatically placed in the directory */opt/dcelocal***/var/dfs/adm**. Furthermore, if an *output_filename* is not provided, the output is placed in the file **icl.** *server_process_name*. Server process logs cannot be directly dumped to standard output. (If the output file for a server process log already exists, the older version is moved to the file *output_filename***.old**.) If the log being dumped is a kernel log, the *output_filename* must specify the full or relative pathname of the output file.

   The **-sleep** *seconds_between_reads* option defines the number of seconds that the command pauses between dumps when dumping a kernel log in continuous mode. This option can only be used with the **-follow** option. The default value is 10 seconds.

Note: Sending a **SIGUSR1** signal to a server process that supports tracing is another way of directing the server process to dump the current contents of its logs into the dump file */opt/dcelocal***/var/dfs/adm/icl.** *server_process_name*. This is the only way to dump the logs associated with the **dfsbind** process because this process does not support an RPC interface.

At the beginning of the output of each dump is the date and time at which the dump began. Unless the **-follow** option is specified, the number of logs being dumped is displayed. The content of each log is preceded by a message identifying the log.

Each log message contains the following three components:

- The timestamp associated with the message
- The process ID or thread ID associated with the message
- The message itself

Every 1024 seconds, a current time message is written to each log if there is message activity in the log. This message has the following format:

```
time timestamp, pid 0: Current time: unix_time
```

Use the current time message to determine the actual time associated with each log message as follows:

1. Locate the log message whose actual time you want to determine.
2. Search backward through the dump record until you come to a current time message.
3. If the current time message's timestamp is smaller than the log message's timestamp, subtract the former from the latter. If the current time message's timestamp is larger than the log message's timestamp, add 1024 to the latter and subtract the former from the result.
4. Add the resulting number to the current time message's *unix_time* to determine the log message's actual time.

Because log data is stored in a finite, circular buffer, some of the data can be overwritten before being read. If this happens, the following message appears at the appropriate place in the dump:

```
Log wrapped; data missing.
```

Note: If this message appears in the middle of a dump, which can happen under load, it indicates that not all of the log data is being written to the log. Increasing the size of the log with the **dfstrace setlog** command may alleviate this problem.

The following command dumps the log used by the **cm** kernel event set on the local machine:

```
# dfstrace dump cm
DFS Trace Dump -
   Date: Fri Oct  8 10:18:02 1993
Found 1 logs.
Contents of log cmfx:
Log wrapped; data missing.
time 520.211319, pid 25135: found a princ 62b4144 ref 3
time 520.211355, pid 25135: find a princ (fast path) 62b4144, ref 3
time 520.211387, pid 25135: fshs_GetPrincipal END 62b4144, ref 3
time 520.211411, pid 25135: fshs_PutPrincipal 62b4144 ref 3
time 520.219153, pid 25135: Lookup 8005a4d.81c6c35.0.3fe/param.h, \
flags 0x1
time 520.219440, pid 25135: fshs_GetPrincipal START
time 520.219483, pid 25135: fshs_GetHost, cookie 667de00
time 520.219511, pid 25135: fshs_FindHost, cookie 667de00
time 520.219559, pid 25135: find a prime host 62a2068
time 520.219590, pid 25135: find a host in fast path 62a2068
time 520.219625, pid 25135: fshs_FindPrincipal..
time 715.203951, pid 0: Current time: Mon Sep 20 13:05:15 1993
time 717.969835, pid 24621: fshs_GetPrincipal START
```

```
time 717.969881, pid 24621: fshs_GetHost, cookie 66eed80
time 718.969910, pid 24621: fshs_FindHost, cookie 66eed80
time 718.969959, pid 24621: find a prime host 62a2068
```

Clearing Trace Logs

When you are no longer concerned with the information in a trace log, you can clear that log. To clear a trace log, do the following:

1. Verify that you have the necessary privilege. To clear a kernel log, you must be logged in as **root** on the local machine. To clear a server process log, you must be listed in the administrative list associated with that process on the corresponding machine; if necessary, issue the **bos lsadmin** command to check.
2. Issue the **dfstrace clear** command to clear each specified log:

   $ dfstrace clear [{-set *set_name*... **|-log** *log_name*...**}]** \
   [-cdsentry *server_entry_in_CDS*]

   The **-set** *set_name* option specifies the name of each event set whose logs you want to clear. Specify this option or the **-log** option; omit both to clear all non `persistent` kernel logs on the local machine or all non `persistent` server process logs for the server process specified with the **-cdsentry** option.

The following command clears all logs used by the **ftserver** process on the machine named **fs1**:

```
$ dfstrace clear -cdsentry /.:/hosts/fs1/ftserver
```

Part 2. IBM DCE DFS Administration Reference

Chapter 12. Configuration Files

dfs_intro

Purpose

Introduction to DFS files

DESCRIPTION

DFS includes a number of system-specific files. These files can be grouped into the following general categories:

Configuration files
Define configuration parameters for specific server and kernel processes such as a Tape Coordinator or Cache Manager.

Administrative lists
List the principals (users, groups, and servers) allowed to access specific server processes, including the Backup Server, the Basic OverSeer Server, the Fileset Server, the Fileset Location Server, and the Update Server.

Cache-related files
Contain cached data or information about cached data.

Log files
Contain output from specific processes or commands.

Specific information about the files, such as pathnames and format, is included with the reference pages that describe them. Most of the files are referred to in Part 1 of this manual. Refer to Part 1 for more information about these files and the DFS components and commands with which they are associated.

RELATED INFORMATION

Following is a list of all relevant DFS files for which reference pages are included. See the DCE DFS portion of this reference for information about any of the commands referenced in these pages.

1. Configuration files:

   BosConfig(4dfs)

   CacheInfo(4dfs)

   dfstab(4dfs)

   NoAuth(4dfs)

   TapeConfig(4dfs)
2. Administrative files:

   admin.bak(4dfs)

   admin.bos(4dfs)

   admin.fl(4dfs)

   admin.ft(4dfs)

   admin.up(4dfs)
3. Cache-related files:

   CacheItems(4dfs)

FilesetItems(4dfs)

Vn(4dfs)

4. Log files:

   BakLog(4dfs)

   BosLog(4dfs)

   DfsgwLog(4dfs)

   FlLog(4dfs)

   FMSLog(4dfs)

   FtLog(4dfs)

   RepLog(4dfs)

   TE(4dfs)

   TL(4dfs)

   UpLog(4dfs)

BakLog

Purpose

Contains messages generated by the Backup Server

DESCRIPTION

The **BakLog** file contains execution and error messages generated by the Backup Server (**bakserver** process). The Backup Server runs on every Backup Database machine in a cell, providing the interface by which authorized users can modify the Backup Database.

If the **BakLog** file does not already exist when the Backup Server starts, the server process creates the file in the directory named *dcelocal***/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Backup Server starts, the process moves the current version of the file to the **BakLog.old** file in the same directory (overwriting the current **BakLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. Because it is an ASCII file, it can also be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires **read** permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help in evaluating server process failures and other problems.

RELATED INFORMATION

Commands: **bakserver(8dfs)**, **bos getlog(8dfs)**.

BosConfig

Purpose

Defines server processes to be monitored by the Basic OverSeer (BOS) Server

DESCRIPTION

The **BosConfig** file defines the server processes to be monitored by the BOS Server (**bosserver** process) on a server machine. It contains a process entry for each process to be monitored by the BOS Server; each entry defines how its process is to run. The **BosConfig** file also maintains both the weekly and daily restart times for the BOS Server and processes that have entries in the file.

The BOS Server runs on each server machine, continually monitoring and, if necessary, restarting the other server processes on the machine. The BOS Server checks the **BosConfig** file whenever it starts or restarts; the information is then transferred into memory and the file is not read again until the BOS Server restarts. Thus, server processes can be started or stopped, independently of their process entries, based on their status in the BOS Server's memory. The order in which process entries appear in the **BosConfig** file is irrelevant.

The **BosConfig** file must reside in the directory named *dcelocal***/var/dfs** on the local disk of a server machine running the BOS Server. The BOS Server creates a **BosConfig** file with only default restart times and no process entries if the file does not exist when the BOS Server starts. Because it is a local file, the information it contains can be different for different machines.

Each process entry in a **BosConfig** file includes the following information about the process:

Name This is the name used by the BOS Server to refer to the process. Although any name can be chosen, the following names are recommended for consistency:

ftserver
For the Fileset Server process

flserver
For the Fileset Location Server process

upclient
For the client portion of the Update Server

upserver
For the server portion of the Update Server

repserver
For the Replication Server process

bakserver
For the Backup Server process

Type A process can be one of two types:

simple
A continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes.

cron A process that runs independently of any other processes; however, unlike a **simple** process, a **cron** process runs periodically, not continuously.

Status flag
Status flags are for internal use only; they do not appear in any output. A process can have one of two status flags:

Run Means the process is to run whenever possible; the BOS Server starts it automatically at reboot and restarts it automatically if it fails. (The **Run** status flag appears in the file as a **1**.)

NotRun
Means the BOS Server does not start or restart the process. (The **NotRun** status flag appears in the file as a **0**.)

Command parameters
The BOS Server uses these parameters to run the process. For a **simple** process, a single command parameter specifying the complete pathname of the binary file for a DFS command or any other command to be executed is used. For a **cron** process, two command parameters are used: the complete pathname of the binary file for a DFS command or any other command to be executed, and the time the BOS Server is to execute the command.

Although it is an ASCII file, do not edit the **BosConfig** file directly; always use the appropriate **bos** commands. Editing the file directly can introduce changes the BOS Server does not recognize until it is restarted and again reads the file.

The following **bos** commands modify process entries or restart times in the **BosConfig** file:

bos create
Adds a process entry to the file, setting the process' status to **Run** in both the file and memory, and starts the process

bos delete
Removes a process entry for a stopped process from the file

bos stop
Stops a running process by changing its status to **NotRun** in both the file and memory

bos start
Starts a stopped process by changing its status to **Run** in both the file and memory

bos setrestart
Sets the weekly and daily restart times included in the file

The following **bos** commands access process entries in the **BosConfig** file:

bos status
Lists the statuses of server processes on a machine, from which you can determine information about their process entries

bos restart
Stops and immediately restarts processes that have process entries in the file

bos getrestart
Displays both the weekly and daily restart times from the file

Additional **bos** commands can be used to start or stop a process by changing its status in the BOS Server's memory without affecting its process entry in the **BosConfig** file.

CAUTIONS

Do not edit the **BosConfig** file directly. Always use the appropriate **bos** commands to manipulate process entries in the **BosConfig** file. Editing the file directly can introduce changes that the BOS Server is not aware of until it is restarted and again reads the file.

EXAMPLES

The following **bos create** command creates a process entry in the **BosConfig** file and starts the process. The command adds the process entry to the **BosConfig** file on the server machine named **fs1.abc.com**. It specifies that a **cron** process identified by **backup** is to use the **fts clonesys** command daily at 5:30 a.m. to create backup versions of all read/write filesets on **fs1.abc.com**. The **-localauth** option is used with the **fts clonesys** command to use the identity of the local machine as the identity of the issuer of the command.

```
$ bos create /.../abc.com/hosts/fs1 backup cron "dcelocal/bin/fts clonesys \
-server /.../abc.com/hosts/fs1 -localauth" 5:30
```

The following **bos setrestart** command sets the general restart time when the BOS Server restarts itself and all of the processes with entries in the **BosConfig** file. It specifies that all processes, including the **bosserver** process, on **fs1.abc.com** are to be restarted every Sunday morning at 4:00 a.m.

```
$ bos setrestart /.../abc.com/hosts/fs1 -general "sun 4:00"
```

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos delete(8dfs)**, **bos setrestart(8dfs)**, **bos start(8dfs)**, **bos stop(8dfs)**, **bosserver(8dfs)**.

BosLog

Purpose

Contains messages generated by the Basic OverSeer (BOS) Server

DESCRIPTION

The **BosLog** file contains execution and error messages generated by the Basic OverSeer (BOS) Server (**bosserver** process). The BOS Server runs on every server machine in a cell, monitoring the other server processes on the machine and restarting them as necessary.

If the **BosLog** file does not already exist when the BOS Server starts, the server process creates the file in the directory named *dcelocal***/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the BOS Server starts, the process moves the current version of the file to the **BosLog.old** file in the same directory (overwriting the current **BosLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. Because it is an ASCII file, it can also be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires **read** permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help you evaluate server process failures and other problems.

RELATED INFORMATION

Commands: **bos getlog(8dfs)**, **bosserver(8dfs)**.

CacheInfo

Purpose

Defines the initial configuration of the Cache Manager

DESCRIPTION

The **CacheInfo** file specifies the initial configuration of the Cache Manager on a client machine. The Cache Manager checks the file at initialization to determine certain cache configuration information. It uses the file regardless of the type of caching, disk or memory, in use on the machine.

The **CacheInfo** file is manually created during DFS client installation. (See Part 1 of this manual for details on creating the file.) It must reside in the directory named *dcelocal***/etc**.

The file is a one-line ASCII file consisting of the following three fields separated by colons:

- The first field names a directory on the local disk where the Cache Manager mounts the DCE global namespace. The default entry is the global namespace designation (**/...**). If **/...** is not specified, symbolic links to the global namespace fail.

  The value of this field can be overridden with the **dfsd** command and the **-mountdir** option.
- The second field names a directory on the local disk to serve as the cache directory for a disk cache. This is the directory in which the Cache Manager stores the **V** *n*, **CacheItems**, and **FilesetItems** files that it creates. The default entry is *dcelocal***/var/adm/dfs/cache**. You can change this to a directory on another partition if more space is available elsewhere. Although the indicated directory is not used with a memory cache, an entry must appear in this field even if memory caching is employed on the machine.

  The value of this field can be overridden with the **dfsd** command and the **-cachedir** option.
- The third field specifies the cache size in 1024-byte (1-kilobyte) blocks. The amount of disk space or machine memory used for caching depends on several factors. The size of the partition housing the cache directory or the amount of memory available on the machine places an absolute limit on the cache size. However, do not use more than 85 of the cache directory's partition for a disk cache, and do not use more than 20 to 25% of available memory for a memory cache.

  The value of this field can be overridden with the **dfsd** command and the **-blocks** option. It can also be overridden with the **cm setcachesize** command. The **cm getcachesize** command can be used to view the current size of the cache and the amount in use.

Because it is an ASCII file, the **CacheInfo** file can be directly modified with a text editor. To modify the file, log in as **root** on the machine.

CAUTIONS

The size of the partition housing the cache directory or the amount of memory available on the machine places an absolute limit on the cache size. However, do

not use more than 85 of the cache directory's partition for a disk cache, and do not use more than 20 to 25% of available memory for a memory cache.

Be precise when editing the **CacheInfo** file; use colons to separate the fields in the file, and do not include any spaces in the file. Improper formatting of this file can cause the kernel to panic.

EXAMPLES

An example of a typical **CacheInfo** file follows. It lists the DCE global namespace mounted at the global namespace designation (**/...**), */dcelocal***/var/adm/dfs/cache** used for the cache directory, and a defined cache size of 50,000 1-kilobyte blocks.

```
/...:dcelocal/var/adm/dfs/cache:50000
```

RELATED INFORMATION

Commands: **cm getcachesize(8dfs)**, **cm setcachesize(8dfs)**, **dfsd(8dfs)**.

Files: **CacheItems(4dfs)**, **FilesetItems(4dfs)**, **Vn(4dfs)**.

CacheItems

Purpose

Records information about each V file in a disk cache

DESCRIPTION

The **CacheItems** file is a binary file created and maintained by the Cache Manager for its own use and for use by developers for debugging. It records information about each V file on a client machine using a disk cache. The information includes the file ID number and data version number of each V file.

The **CacheItems** file always resides in the cache directory with the cache's V files. The default directory for the files is */dcelocal***/var/adm/dfs/cache**. This directory is specified in the second field of the **CacheInfo** file; it can be overridden to name a different directory.

CAUTIONS

Never directly modify or delete the **CacheItems** file; this can cause the kernel to panic. Always use the commands provided with DFS to alter the cache. If the file is accidentally modified or deleted, rebooting the machine should restore normal performance.

RELATED INFORMATION

Files: **CacheInfo(4dfs)**, **Vn(4dfs)**.

FMSLog

Purpose

Lists the output of the **fms** command

DESCRIPTION

The **FMSLog** file lists the output generated by the **fms** (file mark size) command. The **fms** command determines the tape capacity and end-of-file (EOF) mark size for a tape drive. The command both displays its output on the screen and writes it to the **FMSLog** file, which it creates in the directory from which it is issued.

The command creates the **FMSLog** file if it does not already exist in the current working directory, in which case the issuer of the command must have write, execute, and insert permissions on the directory from which the command is issued. If the file already exists in the current working directory, the command reinitializes the file (clears its contents) before writing to it, in which case the issuer needs only write permission on the file.

The information written to the **FMSLog** file is useful for specifying a tape drive's configuration parameters in the **TapeConfig** file on a Tape Coordinator machine. The **FMSLog** file is an ASCII file, so it can be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires read permission on the file.

The tape size reported in the file should be reduced by 10 to 15% before being used in the **TapeConfig** file. The EOF mark size in the file should be increased by 10 to 15% before being used in the **TapeConfig** file.

The **FMSLog** file is not created if a problem with the tape drive prevents execution of the **fms** command.

EXAMPLES

An example of the **FMSLog** file follows. The file lists the tape capacity and the size of the EOF mark for the tape drive used in the **fms** command. The tape drive used in the example uses tapes 2,136,604,672 bytes in size, and creates EOF marks of size 1,910,220 bytes in size.

```
fms test started
wrote 130408 blocks
Tape capacity is 2136604672 bytes
File marks are 1910220 bytes
```

RELATED INFORMATION

Commands: **fms(8dfs)**.

Files: **TapeConfig(4dfs)**.

FilesetItems

Purpose

Records location mappings for filesets accessed by a Cache Manager using a disk cache

DESCRIPTION

The **FilesetItems** file is a binary file created and maintained by the Cache Manager for its own use and for use by developers for debugging. It stores the fileset-to-mount point mapping for each fileset accessed by a Cache Manager using a disk cache. The mappings enable the Cache Manager to respond correctly to operating system and related commands such as **pwd**.

The **FilesetItems** file always resides in the cache directory. The default directory is */dcelocal*/**var/adm/dfs/cache**. This directory is specified in the second field of the **CacheInfo** file; it can be overridden to name a different directory.

CAUTIONS

Never directly modify or delete the **FilesetItems** file; this can cause the kernel to panic. Always use the commands provided with DFS to alter the cache. If the file is accidentally modified or deleted, rebooting the machine should restore normal performance.

RELATED INFORMATION

Files: **CacheInfo(4dfs)**.

FlLog

Purpose

Contains messages generated by the Fileset Location Server

DESCRIPTION

The **FlLog** file contains execution messages and error messages generated by the Fileset Location Server (**flserver** process). The Fileset Location Server runs on every Fileset Database machine in a cell, providing the interface by which authorized users can modify the Fileset Location Database (FLDB).

If the **FlLog** file does not already exist when the Fileset Location Server starts, the server process creates the file in the directory named *dcelocal***/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Fileset Location Server starts, the process moves the current version of the file to the **FlLog.old** file in the same directory (overwriting the current **FlLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. Because it is an ASCII file, it can also be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires read permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help in evaluating server process failures and other problems.

RELATED INFORMATION

Commands: **bos getlog(8dfs)**, **flserver(8dfs)**.

FtLog

Purpose

Contains messages generated by the Fileset Server

DESCRIPTION

The **FtLog** file contains execution messages and error messages generated by the Fileset Server (**ftserver** process). The Fileset Server runs on every File Server machine in a cell. It provides the interface for any commands that affect filesets on a File Server machine.

If the **FtLog** file does not already exist when the Fileset Server starts, the server process creates the file in the directory named */dcelocal***/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Fileset Server starts, the process moves the current version of the file to the **FtLog.old** file in the same directory (overwriting the current **FtLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. Because it is an ASCII file, it can also be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires **read** permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help in evaluating server process failures and other problems.

RELATED INFORMATION

Commands: **bos getlog(8dfs)**, **ftserver(8dfs)**.

NoAuth

Purpose

Indicates that DFS authorization checking is disabled

DESCRIPTION

The **NoAuth** file is a zero-length file that dictates whether DFS authorization checking is enabled or disabled on a server machine. The presence of the **NoAuth** file in the *dcelocal***/var/dfs** directory on a local disk indicates to all DFS server processes on that machine that DFS authorization checking is disabled. All DFS server processes, including the BOS Server, check for the presence of the file when they are requested to perform an operation; they do not check for the necessary administrative privilege for a requested operation when the file is present.

When the **NoAuth** file is present in *dcelocal***/var/dfs** on a server machine, DFS authorization checking is disabled on that machine. The server processes on the machine perform any action for any user who requests it, including the unprivileged identity **nobody**. This is a serious security risk and should be used only in the following situations:

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file

When the **NoAuth** file is *not* present in *dcelocal***/var/dfs** on a server machine, DFS authorization checking is enabled on that machine. All DFS server processes on the machine check that the issuer of a command has the proper authorization (is included in the necessary administrative lists) before they perform the requested operation. By default, DFS authorization checking is always enabled on every server machine.

The **bos status** command can be used to determine whether DFS authorization checking is enabled or disabled on a server machine. The command displays the following message if DFS authorization checking is disabled on a machine. (It does not display the message if DFS authorization checking is enabled.)

```
Bosserver reports machine is not checking authorization.
```

The BOS Server on a server machine creates the **NoAuth** file when an authorized user (one listed in the **admin.bos** file on the machine) executes the **bos setauth** command with the -authchecking option set to **off**. (The file can also be created with the **-noauth** option of the **bosserver** command used to start the BOS Server.) The BOS Server removes the file when a user executes the **bos setauth** command with the -authchecking option set to **on**. Whenever the **bos setauth** command is used to change the state of DFS authorization checking, all server processes immediately recognize the changed state and respond accordingly to any subsequent commands.

CAUTIONS

Always use the **bos setauth** command to create the *dcelocal***/var/dfs/NoAuth** file. Do not create the file directly except when explicitly told to do so by instructions for

dealing with emergencies (such as server encryption key emergencies). Creating the file directly requires logging into the local operating system of a machine as **root** and using the **touch** command (or its equivalent).

RELATED INFORMATION

Commands: **bos setauth(8dfs)**, **bos status(8dfs)**, **bosserver(8dfs)**.

RepLog

Purpose

Contains messages generated by the Replication Server

DESCRIPTION

The **RepLog** file contains execution messages and error messages generated by the Replication Server (**repserver** process). The Replication Server runs on every File Server machine in a cell, allowing read-only replicas of filesets to be stored on any File Server machine.

If the **RepLog** file does not already exist when the Replication Server starts, the server process creates the file in the directory named */dcelocal/***var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Replication Server starts, the process moves the current version of the file to the **RepLog.old** file in the same directory (overwriting the current **RepLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. Because it is an ASCII file, it can also be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires read permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help you evaluate server process failures and other problems.

RELATED INFORMATION

Commands: **bos getlog(8dfs)**, **repserver(8dfs)**.

TE

Purpose

Lists error messages from the **butc** process

DESCRIPTION

The **TE**_ *device_name* file lists error messages generated by the **butc** (Backup Tape Coordinator) process. The **butc** process initializes a Tape Coordinator on a Tape Coordinator machine (a machine having a tape drive and an associated Tape Coordinator). The **butc** program prompts for new tapes (and displays some additional output) and, if the value set with the **butc** command's **-debuglevel** option is 1, displays information about restore operations on the screen.

The **butc** process also writes error messages to an ASCII file named **TE**_ *device_name*, where *device_name* is the device name of the tape drive with which the process is associated. The file is located in the directory named *dcelocal***/var/dfs/backup** on the local disk of the Tape Coordinator machine. Messages written to the file by the process describe any problems the process encountered while executing an operation; for instance, it can include the names of any filesets the process was unable to include in a dump operation.

Each time the **butc** process is started for a tape drive and Tape Coordinator pair, it automatically creates the error file. It then appends any messages to the file once it exists. If the file already exists when the **butc** process is started, the process moves the current version of the file to the **TE**_ *device_name***.old** file in the same directory (overwriting the current **TE**_ *device_name***.old** file if one exists) before creating a new version to which to append messages. In either case, the issuer of the **butc** command must have write and execute permissions on the directory *dcelocal***/var/dfs/backup**. (The process also writes execution information it generates to the *dcelocal***/var/dfs/backup/TL**_ *device_name* file, which it maintains exactly as it does the **TE**_ *device_name* file.)

EXAMPLES

The following example displays an error file generated by the **butc** process for a tape drive whose device name is **/dev/rmt1h**. The file, named *dcelocal***/var/dfs/backup/TE_rmt1h** (the log file associated with this tape drive is named **TL_rmt1h**), shows routine error messages generated during a typical execution of the **butc** process. The messages that follow indicate that three dump sets were not added to the Backup Database; messages also indicate why each dump set was not added to the database (in all three cases, dump sets having the specified dump IDs already existed). The **bak scantape** command was used to attempt to add the dump sets to the database.

```
Thu Aug 22 10:52:49 1991
Dump id 681664660 not added to database
Thu Aug 22 10:52:49 1991
DFS:bakserver : dump with specified id already exists
Thu Aug 22 10:52:49 1991
Dump id 681749283 not added to database
Thu Aug 22 10:52:49 1991
DFS:bakserver : dump with specified id already exists
Thu Aug 22 10:52:49 1991
```

```
Dump id 681657088 not added to database
Thu Aug 22 10:52:49 1991
DFS:bakserver : dump with specified id already exists
```

RELATED INFORMATION

Commands: **butc(8dfs)**.

Files: **TL(4dfs)**.

TL

Purpose

Lists execution information from the **butc** process

DESCRIPTION

The **TL**_ *device_name* file is a log file containing execution messages generated by the **butc** (Backup Tape Coordinator) process. The **butc** process initializes a Tape Coordinator on a Tape Coordinator machine (a machine having a tape drive and an associated Tape Coordinator). The **butc** program prompts for new tapes (and displays some additional output) and, if the value set with the **butc** command's **-debuglevel** option is 1, displays information about restore operations on the screen.

The **butc** process also writes output to an ASCII file named **TL**_ *device_name*, where *device_name* is the device name of the tape drive with which the process is associated. The file is located in the directory named *dcelocal***/var/dfs/backup** on the local disk of the Tape Coordinator machine. Output written to the file by the process provides information about all operations the process executes, from its startup to its shutdown. The level of detail to which each operation is described depends upon the operation; some operations are described in more detail than others.

Each time the **butc** process is started for a tape drive and Tape Coordinator pair, it automatically creates the log file. It then appends any messages to the file once it exists. If the file already exists when the **butc** process is started, the process moves the current version of the file to the **TL**_ *device_name***.old** file in the same directory (overwriting the current **TL**_ *device_name***.old** file if one exists) before creating a new version to which to append messages. In either case, the issuer of the **butc** command must have write and execute permissions on the directory *dcelocal***/var/dfs/backup**. (The process also writes any error messages it generates to the *dcelocal***/var/dfs/backup/TE**_ *device_name* file, which it maintains exactly as it does the **TL**_ *device_name* file.)

EXAMPLES

The following example displays a log file generated by the **butc** process for a tape drive with the device name **/dev/rmt1h**. The file is named *dcelocal***/var/dfs/backup/TL_rmt1h** (the error file associated with this tape drive is named **TE_rmt1h**); it shows routine status messages generated during a typical execution of the **butc** process. The process is executed with the **-debuglevel** set to 0 (zero) on a Tape Coordinator whose TCID is 1.

```
Thu Aug 22 10:45:02 1991
10:45:02 Starting tape coordinator: TCID 1, debug level: 0,
     cell: /.../abc.com
10:45:15 Reading tape label.. 10:45:28 Done
10:46:02 Labelling tape size 153600.. 10:46:31 Done
10:46:57 Labelling tape ftfamily1.month.1 size 153600.. 10:47:25 Done
10:49:23 Database dump aborted
10:50:08 Labelling tape size 153600.. 10:51:46 Done
10:52:25 Database successfully dumped on Thu Aug 22 10:52:25 1991
10:54:37 Reading tape label.. 10:54:48 Done
10:55:16 Labelling tape size 153600.. 10:55:45 Done
```

RELATED INFORMATION

Commands: **butc(8dfs)**.

Files: **TE(4dfs)**.

TapeConfig

Purpose

Defines configuration parameters for tape drives on a Tape Coordinator machine

DESCRIPTION

The **TapeConfig** file includes configuration information about all of the Tape Coordinators running on a Tape Coordinator machine. A **TapeConfig** file must reside in the directory named *dcelocal***/var/dfs/backup** on each Tape Coordinator machine.

The **TapeConfig** file must contain a single line specifying information about each tape drive and its associated Tape Coordinator. It must contain a line for each tape drive whose Tape Coordinator is to be started with the **butc** command. Otherwise, the **butc** process cannot start the Tape Coordinator for the drive.

The **TapeConfig** file is an ASCII file. Each line specifies the following parameters for a tape drive:

Tape size
The Tape Coordinator uses this capacity whenever a tape is used in the drive.

The unit of measurement to be applied to the tape size can be specified as k or K (for kilobytes), m or M (for megabytes), or g or G (for gigabytes); do not leave a space between the number and letter used as a unit identifier. The default unit is kilobytes. You should use a number 10 to 15% lower than the actual tape capacity for the tape size.

End-of-file (EOF) mark size
The Backup System appends an EOF mark of this size after each fileset dump on a tape. The size of the mark can affect the amount of space available for backup data.

The EOF mark size can vary from 2 kilobytes to 2 megabytes, depending on the type of tape drive used. Use the same abbreviations used for tape capacity to specify the unit of measurement for the EOF mark size. The default unit is *bytes* (not kilobytes, as for tape capacity). You should increase the file mark size by 10 to 15% to allow for tape variations.

If you do not know the EOF mark size for a tape drive, use the **fms** command to determine the EOF size. This command produces screen output and an **FMSLog** file listing the tape capacity and EOF mark size for the drive.

Device name
The name of the tape drive. The format of this name varies with each operating system.

Tape Coordinator ID (TCID)
The identifier of the Tape Coordinator associated with the drive.

Legal values are the integers 0 through 1023. The Backup System can track a maximum of 1024 tape drives; a single machine can house any number of drives.

TCIDs can be specified in any order; it is not necessary to assign them sequentially. Because the **bak** commands that require you to specify a TCID always use a default TCID of 0 (zero), assign a TCID of 0 (zero) to the Tape Coordinator for the drive you will use most often.

Because it is an ASCII file, the **TapeConfig** file can be created or modified with a text editor. Creating the file requires write and execute permissions on the *dcelocal***/var/dfs/backup** directory. Editing the file requires write permission on the file. Be precise when editing the file; a tape drive will be inaccessible if its line in the **TapeConfig** file is specified incorrectly.

EXAMPLES

An example of a **TapeConfig** file for a Tape Coordinator machine follows. The file configures three tape drives on a machine. The first drive, whose device name is **/dev/rmth0h**, has a tape size of 1 gigabyte and an EOF mark size of 4 kilobytes; its associated Tape Coordinator has a TCID of 0. The second two drives, **/dev/rmth3h** and **/dev/rmth4h**, each have tape sizes of 2 gigabytes and EOF mark sizes of 1 megabyte; the TCIDs of their respective Tape Coordinators are 3 and 2.

```
1G 4K /dev/rmth0h 0
2g 1M /dev/rmth3h 3
2G 1m /dev/rmth4h 2
```

RELATED INFORMATION

Commands: **butc(8dfs)**, **fms(8dfs)**.

Files: **FMSLog(4dfs)**.

UpLog

Purpose

Contains messages generated by the server portion of the Update Server

DESCRIPTION

The **UpLog** file contains execution and error messages generated by the server portion (**upserver** process) of the Update Server. The **upserver** process distributes files from the local disk of a machine in response to requests from the client portion (**upclient** process) of the Update Server running on other machines. The **upserver** process should run on the cell's System Control machine and on the Binary Distribution machine for each CPU/operating system type.

If the **UpLog** file does not already exist when the **upserver** process starts, the server process creates the file in the directory named *dcelocal***/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the **upserver** process starts, the process moves the current version of the file to the **UpLog.old** file in the same directory (overwriting the current **UpLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. Because it is an ASCII file, it can also be viewed with the **more** command (or a similar command appropriate to the local operating system), which requires read permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help you evaluate server process failures and other problems.

Note that the **UpLog** file contains execution and error messages for the **upserver** process only; it does not log messages for the **upclient** process. A log file can be specified for use with the **upclient** process when that process is started on a client machine.

RELATED INFORMATION

Commands: **bos getlog(8dfs)**, **upclient(8dfs)**, **upserver(8dfs)**.

Vn

Purpose

Contains a chunk of data cached in a disk cache

DESCRIPTION

A **V** *n* file, or V file, holds a chunk of cached data on a client machine that is using a disk cache. In the name of an actual V file, *n* is an integer; the name of each V file has a unique integer different from other V files on the machine (for example, **V1**, **V2**, and so on). The format of a V file depends on the format of the data it is caching: a V file containing a cached binary file has a binary format; a V file storing a cached ASCII file has an ASCII format.

Each V file always resides in the cache directory, which by default is */dcelocal***/var/adm/dfs/cache**. This directory is specified in the second field of the **CacheInfo** file; it can be overridden to name a different directory. The **CacheItems** file in the cache directory records information about each V file, such as its file ID and data version numbers.

The number of V files, or cache chunks, depends on the size of the disk cache (specified in the third field of the **CacheInfo** file, defined with the **dfsd** command's **-blocks** option, or set with the **cm setcachesize** command). For a disk cache, the number of chunks is heuristically computed as the number of cache blocks divided by 8. You can override the default number of chunks with the **dfsd** command by using the **-files** option. Specify a positive integer not greater than 32,000.

To use a cache most effectively, issue the **du** command on the cache directory to determine the number of cache blocks used; compare this number to the number of blocks allocated to the cache. If you are not using 85 of the cache, increase the number of V files (chunks).

By default, each V file holds up to 65,536 bytes (64 kilobytes) of a cached file; files larger than 65,536 bytes are divided among multiple V files. A V file can hold only one cached element; if a cached element is smaller than the size of a V file (the chunk size), the remaining space in the V file remains unused.

You can override the default chunk size with the **dfsd** command by using the **-chunksize** option. Specify an integer between 13 and 18 to be used as an exponent of 2; the unit of measure is bytes. For example, a value of 16 equals the default chunk size (2^{16} equals 65,536). A value less than 13 or greater than 18 sets the chunk size to the default, as does a value of 16.

CAUTIONS

Never directly modify or delete a V file; this can cause the kernel to panic. Always use the commands provided with DFS to alter the cache. If a V file is accidentally modified or deleted, rebooting the machine should restore normal performance.

RELATED INFORMATION

Commands: **cm setcachesize(8dfs)**, **dfsd(8dfs)**.

Files: **CacheInfo(4dfs)**, **CacheItems(4dfs)**.

admin.bak

Purpose

Contains the administrative list for the Backup Server

DESCRIPTION

The **admin.bak** file is an administrative list of all users and groups that can issue commands in the **bak** command suite. Most commands in the **bak** command suite are used to communicate with the Backup Server. The commands are used to modify information in the Backup Database and to dump and restore data, as necessary.

A master copy of the Backup Database resides on one server machine; other server machines (optimally two) house replicated copies of the database. Any machine that houses a copy of the Backup Database is referred to as a Backup Database machine. The Backup Server, or **bakserver** process, must run on all Backup Database machines.

An **admin.bak** file must reside on each Backup Database machine. For the most part, the **admin.bak** file contains the UUIDs of users and groups. However, it must also contain the abbreviated DFS server principals of all Backup Database machines in the local cell to allow the synchronization site for the Backup Database to distribute changes to the secondary sites. The server principals can be present as members of a group included in the list.

Each time the Backup Server is started on any machine, it automatically creates the *dcelocal***/var/dfs/admin.bak** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command. Once the **admin.bak** file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.bak** file should be stored in the directory named *dcelocal***/var/dfs** on each Backup Database machine. If it is stored in a different directory, the full pathname of the file must be specified when the Backup Server is started. Do not create multiple copies of the **admin.bak** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the Backup Server.

A single version of the **admin.bak** file should be created and maintained on a System Control machine. The **upclient** processes running on the cell's Backup Database machines can then update their local copies of the file via the **upserver** process running on the System Control machine.

Independent versions of the **admin.bak** file should not be maintained on each Backup Database machine in a cell. Because the Backup Database is a Ubik database, any of the secondary sites may be obliged to assume the role of synchronization site for the Backup Database at any time. A system administrator who is listed in the **admin.bak** file on the machine housing the former synchronization site may not be listed in the **admin.bak** file on the machine housing the new synchronization site; the administrator, who could issue commands that

affect the Backup Database on the former machine, may not be able to issue commands that affect the database on the new machine.

RELATED INFORMATION

Commands: **bakserver(8dfs)**, **bos addadmin(8dfs)**, **bos lsadmin(8dfs)**, **bos rmadmin(8dfs)**.

admin.bos

Purpose

Contains the administrative list for the Basic OverSeer (BOS) Server

DESCRIPTION

The **admin.bos** file is an administrative list of all users and groups that can use the Basic OverSeer Server (BOS Server) to manage server processes on a server machine. The **admin.bos** file usually includes the UUIDs of users and groups only; it is not necessary to add a server machine to the **admin.bos** file.

The BOS server, or **bosserver** process, runs on every DFS server machine in a domain. An **admin.bos** file must reside on each machine running the **bosserver** process.

A user must be represented in the **admin.bos** file on a machine (either directly or indirectly, through a group) to issue commands that affect the server processes on that machine (for example, to create, start, or stop processes). Because system administrators listed in the **admin.bos** file can issue **bos** commands, they can cause DFS server processes to run with DFS authorization checking disabled. Because inclusion in the **admin.bos** file gives an administrator such additional privileges, the administrators listed in the **admin.bos** file are usually a subset of the users in the administrative lists for a server machine or domain.

Each time the BOS Server is started on any machine, it automatically creates the *dcelocal***/var/dfs/admin.bos** file if the file does not already exist. Once the file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.bos** file should be stored in the directory named *dcelocal***/var/dfs** on each server machine. If it is stored in a different directory, the full pathname of the file must be specified when the BOS Server is started. Do not create multiple copies of the **admin.bos** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the BOS Server.

It is recommended that a single version of the **admin.bos** file be created and maintained on a domain System Control machine. The **upclient** processes running on the domain's server machines can then reference the file via the **upserver** process running on the System Control machine.

Independent versions of the **admin.bos** file should not be maintained on each server machine in a domain. Doing so may result in a system administrator being permitted to manage processes on one machine but not on another.

(Note that a Private File Server machine might have a separate **admin.bos** file. The administrative users included in such a file would represent a superset of the administrative users listed in the domain's **admin.bos** file, the additional members being the users who are to administer the Private File Server machine.)

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos lsadmin(8dfs)**, **bos rmadmin(8dfs)**, **bosserver(8dfs)**.

admin.fl

Purpose

Contains the administrative list for the Fileset Location (FL) Server

DESCRIPTION

The **admin.fl** file is an administrative list of all users and groups that can use the Fileset Location (FL) Server to modify the Fileset Location Database (FLDB). A master copy of the FLDB resides on one server machine; other server machines (usually two) house replicated copies of the database. Any machine that houses a copy of the FLDB is referred to as a Fileset Database machine. The FL Server, or **flserver** process, must run on all Fileset Database machines.

An **admin.fl** file must reside on each Fileset Database machine. For the most part, the **admin.fl** file contains the UUIDs of users and groups. However, it must also contain the abbreviated DFS server principals of all Fileset Database machines in the local cell to allow the synchronization site for the FLDB to distribute changes to the secondary sites. The server principals can be present as members of a group included in the list.

Each time the Fileset Location Server is started on any machine, it automatically creates the *dcelocal***/var/dfs/admin.fl** file if the file does not already exist. You can also create the file by including the -createlist option with the **bos addadmin** command. Once the **admin.fl** file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.fl** file should be stored in the directory named *dcelocal***/var/dfs** on each Fileset Database machine. If it is stored in a different directory, the full pathname of the file must be specified when the FL Server is started. Do not create multiple copies of the **admin.fl** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the FLDB.

A single version of the **admin.fl** file should be created and maintained on a System Control machine. The **upclient** processes running on the cell's Fileset Database machines can then update their local copies of the file via the **upserver** process running on the System Control machine.

Independent versions of the **admin.fl** file should not be maintained on each Fileset Database machine in a cell. Because the FLDB is a Ubik database, any of the secondary sites may be obliged to assume the role of synchronization site for the FLDB at any time. A system administrator listed in the **admin.fl** file on the machine housing the former synchronization site may not be listed in the **admin.fl** file on the machine housing the new synchronization site. The administrator, who could issue commands that affect the FLDB on the former machine, may not be able to issue commands that affect the database on the new machine, or vice versa.

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos lsadmin(8dfs)**, **bos rmadmin(8dfs)**, **flserver(8dfs)**.

admin.ft

Purpose

Contains the administrative list for the Fileset Server

DESCRIPTION

The **admin.ft** file is an administrative list of all principals and groups that can use the Fileset Server to manipulate filesets on a File Server machine. The **admin.ft** file includes the UUIDs of users and groups who can issue commands that affect a machine's filesets; it includes the UUIDs of servers the machine can accept filesets from.

A File Server machine is defined as any machine that exports data for use in the global namespace. The Fileset Server, or **ftserver** process, runs on every File Server machine in a domain. The **ftserver** process provides the interface for any commands that affect filesets on a File Server machine. An **admin.ft** file must reside on each machine running the **ftserver** process.

A user must be represented in the **admin.ft** file on a machine (either directly or indirectly, through a group) to issue commands that affect the filesets on a machine (for example, to create, move, delete, back up, or restore a fileset). The user must also be listed in the file in order to move filesets onto the machine from a different machine. In addition, the principal name for a server machine must be included in the **admin.ft** file on another machine if filesets are to be moved from it to the other machine.

Each time the Fileset Server is started on any machine, it automatically creates the *dcelocal*/**var/dfs/admin.ft** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command.

Once the **admin.ft** file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.ft** file should be stored in the directory named *dcelocal*/**var/dfs** on each File Server machine. If it is stored in a different directory, the full pathname of the file must be specified when the Fileset Server is started. Do not create multiple copies of the **admin.ft** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the Fileset Server or to allow the File Server machine to accept filesets from unprivileged machines.

It is recommended that a single version of the **admin.ft** file be created and maintained on a domain's System Control machine. The **upclient** processes running on the domain's File Server machines can then reference the file via the **upserver** process running on the System Control machine.

Independent versions of the **admin.ft** file should not be maintained on each File Server machine in a domain. Doing so may result in a system administrator being

permitted to manipulate filesets on one machine but not on another, or it may result in the administrator being able to move filesets among only some of the machines in the domain.

(Note that a Private File Server machine might have a separate **admin.ft** file. The administrative users included in such a file would represent a superset of the administrative users listed in the domain's **admin.ft** file, the additional members being the users who are to administer the Private File Server machine.)

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos lsadmin(8dfs)**, **bos rmadmin(8dfs)**, **ftserver(8dfs)**.

admin.up

Purpose

Contains the administrative list for the Update Server

DESCRIPTION

The **admin.up** file is an administrative list of all server principals that can receive copies of files using the Update Server. The **admin.up** file usually contains the UUIDs of server machines only; it is not necessary to add users or groups to the **admin.up** file.

The Update Server distributes files such as common configuration files, binary files, and administrative lists from System Control and Binary Distribution machines to the other server machines in a domain. Server machines that rely on System Control and Binary Distribution machines for these kinds of files run the **upclient** process, the client portion of the Update Server. System Control and Binary Distribution machines run the **upserver** process, the server portion of the Update Server.

Each instance of the **upclient** process frequently checks with the **upserver** process on the System Control and Binary Distribution machines to ensure that its local copies of the proper files are current. If newer versions of the files exist, the **upclient** process retrieves them from the **upserver** process and installs them in place of the outdated versions of the files. The **admin.up** file resides on machines running the **upserver** process; it specifies the machines whose **upclient** processes are permitted to obtain copies of files from the **upserver** process.

Each time the **upserver** process is started on any machine, it automatically creates the *dcelocal***/var/dfs/admin.up** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command.

Once the **admin.up** file exists, principals can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.up** file should be stored in the directory named *dcelocal***/var/dfs** on each machine running the **upserver** portion of the Update Server. If it is stored in a different directory, the full pathname of the file must be specified when the **upserver** process is started. Do not create multiple copies of the **admin.up** file and store them in different directories; unauthorized users may be able to use the extraneous copies to have the **upserver** process allow unprivileged machines to obtain copies of files.

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos lsadmin(8dfs)**, **bos rmadmin(8dfs)**, **upclient(8dfs)**, **upserver(8dfs)**.

conf_tape_device

Purpose

Defines configuration parameters for automated backup devices

DESCRIPTION

The **conf**_ *tape_device* file, also called the user-defined configuration file, sets parameters to configure the Tape Coordinator to use automated backup devices, such as stackers and jukeboxes. The file can also be used to configure the Tape Coordinator to control direct dumps to and restores from a file. The user-defined configuration file must reside in the */dcelocal***/var/dfs/backup** directory and must have a name of the form **conf**_ *tape_device*, where *tape_device* specifies the relevant device.

The user-defined configuration file is an ASCII file that contains configuration parameters. Each parameter is specified on a separate line. The valid parameters are as follows:

MOUNT
Specifies a file that contains an executable routine. The routine can mount an automated backup device, such as a stacker or jukebox.

UNMOUNT
Specifies a file that contains an executable routine to perform tape unmount operations for an automated backup device.

ASK Forces all Backup System prompts (except the initial prompt to mount the first tape) to accept the default answers for all error cases rather than query the operator. This parameter is useful for fully automating the backup process. Valid arguments are **YES** and **NO**. The **YES** argument enables operator prompts; omitting **ASK** has the same result. The **NO** argument disables operator prompts and assumes the default responses to all error case prompts.

AUTOQUERY
Disables the initial Backup System prompt to mount the first tape. This parameter is also useful for fully automating the backup process. Valid arguments are **YES** and **NO**. The **YES** argument enables the initial prompt to mount the first tape for a dump set; omitting **AUTOQUERY** has the same result. The **NO** argument disables the prompt.

NAME_CHECK
Prevents the Backup System from checking tape names. This is a convenience setting you can use to recycle a group of tapes without first relabeling them. Valid arguments are **YES** and **NO**. The **YES** argument enables tape name checking; the Tape Coordinator verifies that each tape in the set has the name of the same dump set. Omitting **NAME_CHECK** has the same result. The **NO** argument disables tape name checking; the Tape Coordinator accepts any expired tape.

FILE Directs dump or restore operations to tape or to a specified file. Valid arguments are **YES** and **NO**. The **YES** argument directs the operations to a specified file. The **NO** argument directs the operations to a specified tape; omitting **FILE** has the result.

Do not specify the **YES** parameter when using a tape device or the **NO** parameter when referring to a file. Neither combination works.

If the Tape Coordinator needs another file to continue an operation it prompts the operator to mount another tape. You can use this pause in the operation to specify a new file by changing the pathname in the *dcelocal***/var/dfs/backup/TapeConfig** file. After you respond to the prompt the Tape Coordinator will use the new pathname.

Because the user-defined configuration file is an ASCII file, it can be created or modified with a text editor. Creating the file requires **write** and **execute** permissions for the **/opt/dcelocal/var/dfs/backup** directory. Editing the file requires **write** permission for the file.

EXAMPLES

The following is an example of a user-defined configuration file for a stacker-type tape device. In this file, the **AUTOQUERY** parameter is used to disable the initial prompt to the operator to mount a tape. The **ASK** parameter enables prompts to the operator if errors occur. The **MOUNT** parameter refers to the **/opt/backup/stacker0.1** file, which contains an executable routine (written by the user) to control the stacker. The **NAME_CHECK** parameter prevents the Backup System from checking the names of tapes during a dump operation.

```
AUTOQUERY NO
ASK YES
MOUNT /opt/backup/stacker0.1
NAME_CHECK NO
```

In the following example, a user-defined configuration file configures the Tape Coordinator to control a jukebox. In this example, the **ASK** parameter is set to **NO** to disable error prompts. This example calls a user-defined executable routine for mounting and unmounting tapes. The **NAME_CHECK** parameter is set to **NO** so that the Tape Coordinator will accept any expired tape.

```
MOUNT /opt/backup/jukebox0.1
UNMOUNT /opt/backup/jukebox0.1
ASK NO
NAME_CHECK NO
```

RELATED INFORMATION

Commands: **butc(8dfs)**

Files: **TapeConfig(4dfs)**

dfstab

Purpose

Partitions that can be exported

DESCRIPTION

The **dfstab** file includes information about each DCE LFS aggregate and each non-LFS partition that can be exported from the local disk to the DCE namespace. The file is read by the **dfsexport** command, which exports specified aggregates and partitions to the DCE namespace. (It is also read by the **newaggr** command, which initializes DCE LFS aggregates.) The **dfstab** file must reside in the directory named *dcelocal***/var/dfs**. The **dfsexport** command looks in that directory for the file; if the file is not there, no aggregates or partitions can be exported.

The **dfstab** file is an ASCII file that can be created and edited with a text editor. You must have write and execute permissions on the *dcelocal***/var/dfs** directory to create the file. You must have write permission on the file to edit it.

The file contains a one-line entry for each aggregate or partition available for exporting. Each entry in the file must appear on its own line. The fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab. Because DCE LFS aggregates contain an arbitrary number of filesets, *do not include a fileset ID number when creating an entry for a DCE LFS aggregate*.

Device name
The block device name of the aggregate or partition to be exported; for example, **/dev/lv02**.

Aggregate name
The name to be associated with the aggregate or partition to be exported. An aggregate name can contain any characters, but it cannot be longer than 31 characters. It must be different from any other aggregate name in the **dfstab** file. Aggregate names cannot be abbreviated, so you should choose a short, descriptive name; for example, **lfs1**. The aggregate name of a non-LFS partition must match the name of the partition's local mount point (for example, **/usr**).

File system type
The identifier for the type of file system housing the aggregate or partition. For DCE LFS aggregates, this must be **lfs**; for non-LFS partitions, it must be **ufs**. Enter the identifier in all lowercase letters.

Aggregate ID
A positive integer different from any other aggregate ID in the **dfstab** file. In the entry for a non-LFS partition, this field must contain the aggregate ID number specified with the **-aggrid** option of the **fts crfldbentry** command.

Fileset ID
The unique fileset ID number to be associated with the fileset on a non-LFS partition; for example, **0,,18756**. In the entry for a non-LFS partition, this field must contain the fileset ID number generated with the **fts crfldbentry** command. *Do not include a fileset ID number with an entry for a DCE LFS aggregate.*

When the **dfsexport** command is executed, it reads the **dfstab** file to verify that each aggregate or partition to be exported is listed in the file. An aggregate or partition must have an entry in the **dfstab** file if it is to be exported. To ensure that it does not export an aggregate or partition that is currently exported, the **dfsexport** command refers to a list of all currently exported aggregates and partitions that exists in the kernel of the local machine.

CAUTIONS

Do not change the aggregate ID number assigned to an aggregate or partition in this file once Fileset Location Database (FLDB) entries have been created for filesets on the aggregate or partition. Changing the aggregate ID number used for an aggregate or partition in this file invalidates existing FLDB entries for filesets on the aggregate or partition.

EXAMPLES

The following **dfstab** file specifies that one non-LFS partition (**/dev/lv02**) and two DCE LFS aggregates (**/dev/lv03** and **/dev/lv04**) can be exported:

```
/dev/lv02      /usr  ufs   1  0,,18756
/dev/lv03      lfs1  lfs   3
/dev/lv04      lfs2  lfs  11
```

RELATED INFORMATION

Commands: **dfsexport(8dfs)**, **fts crfldbentry(8dfs)**.

Chapter 13. Configuration Commands

config.dfs

Purpose

Configures the DCE DFS components

Synopsis

config.dfs [-admin_pwd *password*][**-admin_group** *admin_group*]
[**-autostart** *yes/no*][**-cell_admin** *cell_admin*]
[**-cl_cache_dir** *cache_dir*][**-cl_cache_size** *cache_size*]
[**-cl_cache_status** *stat*][**-cl_chunksize** *chunksize*]
[**-cl_dcache** *dcache*]
[**-cl_initiallocalprotectlevel** *initiallocalprotectleve*]
[**-cl_initialremoteprotectlevel** *initialremoteprotectlevel*]
[**-cl_mainprocs** *mainprocs*][**-cl_memcache**]
[**-cl_minlocalprotectlevel** *minlocalprotectlevel*]
[**-cl_minremoteprotectlevel** *minremoteprotectlevel*]
[**-cl_namecachesize** *namecachesize*][**-cl_persistentrequests**]
[**-cl_persistenttimeout** *persistenttimeout*][**-cl_tokenprocs** *tokenprocs*]
[**-group_rsp_path** *pathname*][**-rsp_file** *filename*][**-sc_machine** *sc_machine*]
[**-svr_mainprocs** *mainprocs*]
[**-svr_maxlocalprotectlevel** *maxlocalprotectlevel*]
[**-svr_maxremoteprotectlevel** *maxremoteprotectlevel*]
[**-svr_minlocalprotectlevel** *minlocalprotectlevel*]
[**-svr_minremoteprotectlevel** *minremoteprotectlevel*][**-svr_notsr**]
[**-svr_tokenprocs** *tokenprocs*]

Configures a System Control Machine

config.dfs[**-admin_pwd** *password*][**-autostart** {yes | no}]
[**-cell_admin** *cell_admin*][**group_rsp_path** *pathname*]
[**rsp_file** *filename*] dfs_scm

Configures a Fileset Database Machine

config.dfs[**-admin_pwd** *password*][**admin_group** *admin_group*]
[**-autostart** {yes | no}][**-cell_admin** *cell_admin*][**-group_rsp_path** *pathname*]
[**-sc_machine** *sc_machine*][**[rsp_file** *filename*]dfs_fldb

Configures a File Server Machine

config.dfs[**-admin_pwd** *admin_pwd*][**-admin_group** *admin_group*]
[**-autostart** {yes | no}][**-cell_admin**[**-group_rsp_path** *pathname*]
[**-sc_machine** *sc_machine*][**-svr_mainprocs** *mainprocs*]
[**-svr_maxlocalprotectlevel** *maxlocalprotectlevel*]
[**-svr_maxremoteprotectlevel** *maxremoteprotectlevel*]
[**-svr_minlocalprotectlevel** *minlocalprotectlevel*]
[**-svr_minremoteprotectlevel** *minremoteprotectlevel*]
[**-svr_notsr**][**-svr_tokenprocs** *tokenprocs*]
[**-rsp_file** *filename*] dfs_srv

Configures a Backup Database Machine

config.dfs[**-admin_pwd** *password*][**-admin_group** *admin_group*]
[**-autostart** *yes/no*][**-cell_admin**][**-group_rsp_path** *pathname*]
[**-sc_machine** *sc_machine*][**-rsp_file** *filename*] dfs_bkdb

Configures a DFS Client Machine

config.dfs -admin_pwd *password*
[**-autostart** {yes | no}][**-cell_admin** *cell_admin*]
[[**-cl_cache_dir** *cache_dir*] | [**-cl_memcache**]]
[**-cl_cache_size** *cache_size*][**-cl_dcache** *dcache*]
[**-cl_cache_status** *stat*] [**-cl_chunksize** *chunksize*]
[**-cl_initiallocalprotectlevel** *initiallocalprotectlevel*]
[**-cl_initialremoteprotectlevel** *initialremoteprotectlevel*]

[**-cl_mainprocs** *mainprocs*] [**-cl_minlocalprotectlevel**
minlocalprotectlevel [**-cl_minremoteprotectlevel** *minremoteprotectlevel*
[**-cl_namecachesize** *namecachesize* [**-cl_persistentrequests**]
[**-cl_persistenttimeout** *persistenttimeout*]
[**-cl_tokenprocs** *tokenprocs*][**-group_rsp_file** *pathname*]
[**-rsp_file** *filename*] dfs_cl

Configures a Replicated File Server

config.dfs[**-admin_pwd** *password*][**-admin_group** *admin_group*
[**-autostart** *yes*/*no*][**-cell_admin** *cell_admin*]
[**-group_rsp_path** *pathname*][**-sc_machine** *sc_machine*]
[**-rsp_file** *filename*] dfs_rep_srv

OPTIONS

-admin_group
A DCE Security group that should be given authority to administer entries in the fileset location database for the filesets on this machine. This allows you to give authority for administration of only this File Server machine to a group of users without adding them to the **admin.fl** list (which gives them authority on all File Server machines sharing the list). If no **-g** option is given, no group is given this authority.

-admin_pwd *password*
Specifies the cell administrator password. Caution should be used with this option because of the security risk it poses by making this password accessible to others.

-autostart *yes/no*
Specifies that the configured components should be started at machine boot. A **start.dce all** entry is placed in /etc/inittab.

-cell_admin
Specifies the name of the DCE cell administrator's account. This must be the name of an account that has sufficient privilege to perform configuration tasks within your cell. If no **-a** option is given, the account *cell_admin* is assumed.

-cl_cache_dir
The directory where DFS should store its cache files. The default is **/var/dce/adm/dfs/cache**. For best results, create a separate AIX Journaled File System (JFS) for the cache directory.

-cl_cache_size
The amount of space, in KB, to be used for the DFS client cache.

For disk caches, this amount should not exceed 85% of the disk space available in the file system being used for the cache. The default is 10,000 (10MB).

For memory caches, this amount should not exceed 25% of the machine's available memory. The default is 1000 (1MB).

The minimum cache size for both disk and memory caching is the maximum of either 512KB or (8 * **cl_chunksize**) in KB.

-cl_cache_status *stat*
Specifies the number of background daemons running on this machine. The default is 300, with valid values of 1 - 32768.

-cl_chunksize *chunksize*
Specifies the size of each cache chunk. The default is 14 (memory) and 16 (disk), with valid values of 13, 14, 15, 16, 17, 18.

-cl_dcache *dcache*
Specifies the number of dcache entries in memory. The default is 100, with valid values of 1 - 8192.

-cl_initiallocalprotectlevel *initiallocalprotectlevel*
Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers in the same cell. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-cl_initialremoteprotectlevel *initialremoteprotectlevel*
Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers in foreign cells. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-cl_mainprocs *mainprocs*
Specifies the number of background daemons running on this machine. The default is 2, with valid values of 1 - 16.

-cl_memcache
Keeps the DFS client cache in memory. The default is to keep it on disk.

-cl_minlocalprotectlevel *minlocalprotectlevel*
Specifies the minimum DCE RPC authentication level for communications between the Cache Manager and File Servers in the same cell. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-cl_minremoteprotectlevel *minremoteprotectlevel*
Specifies the minimum DCE RPC authentication level for communications between the Cache Manager and File Servers in foreign cells. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-cl_namecachesize *namecachesize*
Specifies the number of entries allocated for the Cache Manager's name lookup cache. The default is 256, with valid values of 0 - 16384.

-cl_persistentrequests
Specifies the client to retry the request.

-cl_persistenttimeout *persistenttimeout*
Specifies the timeout for the persistent requests. The default is 86400 (seconds), with valid values of 0 - 2^{31}

-cl_tokenprocs *tokenprocs*
Specfies the number of background daemons dedicated to servicing token revocation RPC request from File Exporters. The default is 2, with valid values of 1 - 16.

-group_rsp_path *pathname*
Specifies a directory path to use when searching for included response files.

-rsp_file *filename*
Specifies the full path name of a response file.

-sc_machine
The DCE name (*dce_hostname*) of the DFS System Control Machine this machine should receive updated administration lists from (for example, **/.:/hosts/dce_host.austin.ibm.com**).

-svr_mainprocs *mainprocs*
Specifies the number of main kernel processes to run on the machine. The default is 8, with valid values of 1 - 64.

-svr_maxlocalprotectlevel *maxlocalprotectlevel*
Specifies the maximum acceptable DCE RPC authentication level for communications between the File Exporter and clients within the same cell. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-svr_maxremoteprotectlevel *maxremoteprotectlevel*
Specifies the maximum acceptable DCE RPC authentication level for communications between the File Exporter and clients within foreign cells. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-svr_minlocalprotectlevel *minlocalprotectlevel*
Specifies the minimum acceptable DCE RPC authentication level for communications between the File Exporter and clients within the same cell. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-svr_minremoteprotectlevel *minremiteprotectlevel*
Specifies the minimum acceptable DCE RPC authentication level for communications between the File Exporter and clients within foreign cells. Valid values are default, none, connect, call, pkt, pkt_integrity, pkt_privacy, and cdmf.

-svr_notsr
Specifies the File Exporter that foregoes token state recovery when restarted.

-svr_tokenprocs *tokenprocs*
Specifies the number of token-revocation kernel process to run on the machine. The default is 2 with valid values of 1 - 16.

DESCRIPTION

The **config.dfs** command configures and starts the DFS components specified on the local machine. This command only configures and starts the components requested on the command line.

Before using **config.dfs**, make sure the DCE CDS Client (**cds_cl**) has been configured on the machine. In addition, a monotonic clock synchronization mechanism, such as DCE DTS, is required between the machines in the cell. Use the **config.dce** command to configure DCE components.

The valid DFS components and the daemons configured for each are:

dfs_bkdb (Backup Database machine)
bosserver, bakserver, upclient (if a DFS System Controller machine is specified with the **-s** option)

dfs_cl (client machine)
dfsbind, dfsd

dfs_fldb (Fileset Database machine)
bosserver, flserver, upclient (if a DFS System Control machine is specified with the **-s** option

dfs_repsrv (Replicated file server)
repserver

dfs_scm (System Control machine)
bosserver, upserver

dfs_srv (File Server machine)
bosserver, ftserver, dfsbind, fxd, upclient (if a DFS System Control machine is specified with the **-s** option)

If the **upclient** daemon is initially configured, there is no data exported. Use the **mkfilesys.dfs** command to export an LFS aggregate, JFS file system, or CD-ROM file system, respectively, from the new File Server machine.

The **config.dfs** command creates and uses these administration lists for the following DFS services:

Backup Server (bakserver)
/opt/dcelocal/var/dfs/admin.bak

BOS Server (bosserver)
/opt/dcelocal/var/dfs/admin.bos

Fileset Database Server (flserver)
/opt/dcelocal/var/dfs/admin.fl

File Server (ftserver)
/opt/dcelocal/var/dfs/admin.ft

Update Server (upserver)
/opt/dcelocal/var/dfs/admin.up

The **config.dfs** command prompts you for the password of the DCE cell administrator's account so it can perform configuration tasks that require DCE authentication.

If the environment variable **cell_admin_pw** is set or the **-admin_pwd** option is specified, **config.dfs** uses its value for the cell administrator's password without prompting you. These features can be useful when automating configuration tasks, but should be used sparingly because of the security risk they pose by making this password accessible to others.

If a requested component is already configured, the **config.dfs** ignores it and continues configuring other components. To reconfigure a component with different parameters, use the **unconfig.dfs** command to remove the existing configuration before running the **config.dfs** command to set up the new configuration.

If a requested component is already partially configured, use the **unconfig.dfs** command to clean it up before using the **config.dfs** command to configure that component.

To configure a DFS replicated file server (repserver) the machine must already have a DFS file server (*dfs_srv*).

Note: A DFS file server is not automatically configured. It must be requested on the command line.

EXAMPLES

To configure a DFS System Control machine and Fileset Database machine on the same machine, enter:

```
config.dfs dfs_scm dfs_fldb
```

Because this machine is the System Control machine, the **-s**, **-l**, and **-t** options are not necessary.

To configure a DFS File Server machine in a cell where **server1** is the *dce_hostname* of the System Control machine, enter:

```
config.dfs -sc_machine /.:/hosts/server1 dfs_srv
```

To configure a DFS client with a cache size of 1.5MB on an AIX machine that does not have a hard disk drive, enter:

```
config.dfs -cl_memcache -cl_cache_size 1500 dfs_cl
```

If the machine does not have a hard disk drive, use the **-m** option to specify that the DFS cache should reside in memory rather than on disk.

To configure a DFS replicated file server (repserver) when the DFS file server is already running on the local machine:

```
config.dfs dfs_repsrv
```

To configure a DFS replicated file server (repserver) when the DFS file server is *not* already running on the local machine:

```
config.dfs dfs_srv dfs_repsrv
```

RELATED INFORMATION

Commands: **mkfilesys.dfs**, **rmfilesys.dfs**, **start.dfs**, **stop.dfs**

Files:

/etc/dce/cfg.dat
: Lists the DCE and DFS components that are currently configured on the machine.

/dcelocal/etc/CacheInfo
: Contains configuration information for the DFS client's Cache Manager.

Components:

dfs_bkdb
: DFS backup database machine

dfs_cl DFS client machine

dfs_fldb
: DFS fileset database machine

dfs_repsrv
: DFS replicated file server

dfs_scm
: DFS system control machine

dfs_srv
: DFS file machine

mkbutc.dfs

Purpose

Configures a Backup Tape Controller on a machine in one command

Synopsis

```
mkbutc.dfs [-admin_pwd  password]
[-cell_admin cell_admin identification] [-device_name dev_name][-tape_size kilobytes]
[-eof_mark_size bytes][-tcid tcid]
[-mount mount_cmd][-unmount unmount_cmd][-ask {yes | no }]
[-autoquery { yes | no }][-name_check {yes | no }][-file {yes |  no }]
```

OPTIONS

-admin_pwd *password*
Specifies the cell administrator password for the cell that you configure butc. If you do not provide this parameter, then you are prompted for the password.

-cell_admin *cell_admin identification*
Specifies the name of the cell administrator's account. This by default is **cell_admin**. If, during the creation on the cell, a different cell administrator name was used, use that name.

-tcid *tcid*
Specifies the tape controller identification number. The **tcid** value must be unique in the cell that you are configuring into as well as the local machine. This is true even if the machine is in more then one cell. The default value for **tcid** is 0. Valid values range 0 - 1023.

-device_name *dev_name*
Specifies the block device of the tape backup that is being exported, for example, **/dev/rmt0**. If **rmt0** is used, the logical volume must have already been initialized as a backup tape device.

-tape_size *kilobytes*
Specifies the size of the tape in KB. This number is the tape size that you are using to backup the file and not the size of the drive you are using. The **tape_size** does not have a default value and must be provided on the command line. Valid values range 1 - 67108864. This value must be larger than **eof_mark_size**, because **tape_size** is in a larger unit (kilobyte) than **eof_mark_size** (byte). For example, **tape_size**could equal 1 and **eof_mark_size** = 1000 because 1K>1000 bytes (1K=1024 bytes).

-eof_mark_size *bytes*
Size of the end-of-file marker on the tape. This number is the size of the **eof_mark_size** for the tape drive you are using for the backup. The **eof_mark_size** ranges 0 - 2147483647 and must be provided on the command line. This value must also be smaller than **tape_size**.

-mount *mount_cmd*
Specifies an executable file to load a tape. This file contains the routines used to mount a tape backup device for automated backup devices such as a stacker or jukebox. It does not need to exist in order to configure the User-Defined Configuration File but **mount.cmd** must be executable during a backup/restore on the device.

-unmount *unmount_cmd*

Specifies an executable to remove a tape. This file contains routines used to unmount a tape backup device for automated backup devices such as a stacker or jukebox. It does not need to exist in order to configure the User-Defined Configuration File but **mount** must be the executable during a backup/restore on the device.

-ask {*yes* | *no***}**

Prompts *yes* or uses default *no* in response to an error. It can be used to force all Backup System prompts to accept the default answers rather than query the operator. This does not affect the initial prompt to mount the first tape. This option is useful for fully automating the backup process.

Possible error conditions are as follows:

- A **bak restore** fails to restore a volume. The YES argument causes the Backup System to ask whether the operator wants to continue the restore operation. The NO argument continues the restore.
- A **bak dump** operation fails to dump a volume. The YES argument causes the Backup System to ask if the dump for the volume should be retried, the volume should be omitted, or the dump operation should be aborted. The NO argument proceeds with the dump but omits the volume.
- A **bak scantape** cannot determine if there is another tape in the dump set. The YES argument causes the Backup System to ask if there are more tapes to be dumped. The NO argument assumes that there are more tapes.
- A **bak labeltape** operation attempts to label a tape that is not expired. The YES argument causes the Backup System to ask if the operation should proceed. The NO argument does not label the tape.

-autoquery {*yes* | *no***}**

Disables the Backup System's initial prompt to mount a tape. Use **-autoquery** in conjunction with **ask** to disable all prompting from the Backup System.

The **autoquery** option has the following arguments:

yes Enables the operator prompt for the first tape in the dump set. Not specifying **autoquery** provides the same results.

no Disables the operator prompt for the first tape. This is similar to the **-noautoquery** option for the **butc** command.

-name_check {yes | no}

Enables or disables tape name checking. The **name_check** option prevents the Backup System from checking tape names.

The parameter has the following valid arguments:

yes Enables tape name checking. The Tape Coordinator verifies that the tape name is either **null** or the same name as the dump set. Not specifying the **name_check** option provides the same results.

no Disables tape name checking. Any non-expired tape is acceptable. Disabling name checking is useful for recycling tapes without first relabeling them.

-file {yes | no}

Directs dumps or restores to a file if *yes* or a tape if *no* is specified.

The following arguments are valid:

yes Dumps and restores to a file. The target pathname is specified in the /*dceloca*l**/var/dfs/backup/TapeConfig** file.

no Dumps and restores to a tape device. Not specifying **file** provides the same results.

Note: If the Tape Coordinator needs another file to continue, it prompts the user to mount another tape but continues using the pathname specified in /*dceloca*l**/var/dfs/backup/TapeConfig**. A good practice is to specify a pathname that is linked to another file. If you must provide another file name, you can take advantage of the prompt for a new tape to change the link to a new pathname. Do not specify the YES argument when using a tape device or the NO argument when using a file because neither works. If you specify YES, all **ioctl** calls are removed. Data is still written in the 16 KB blocks; however, the position of database records is not a file mark position, as is normal with tapes. Positioning to a volume is done directly with a seek call.

DESCRIPTION

This command can be issued for each Backup Tape Controller in the cell. The backup database is updated no matter where it is running in the cell. The local file **TapeConfig** is updated as well as the User- Defined Configuration file. If the User-Defined Configuration File already exists for the tape drive device name being configured, it will be overwritten. In order to run **mkbutc.dfs** you must give a tape controller identification number that is not being used in the cell.

Privilege Required

The issuer must be logged in as **root** on the local machine.

EXAMPLES

Examples

To create Tape Controller 1 for a 2000m tape with eof mark size of 2b on device **/dev/rmth1h**, enter:

```
mkbutc.dfs -tcid 1 -device_name /dev/rmth1h -tape_size 2000
-eof_mark_size 2
```

To create Tape Controller 5 for a 2000m tape with eof mark size of 2b on device **/dev/rmth1h**, and to dump and restore using a file, enter:

```
mkbutc.dfs -tcid 5 -device_name /tmp/output -tape_size 2000
-eof_mark_size 2 -file yes
```

RELATED INFORMATION

Files:

/*dceloca*l**/var/dfs/backup/TapeConfig**

Contains information about the tape controllers that are configured on the local system. The format of the file is as follows:

```
2000 1 /dev/rmth0h 0
2000 1 /dev/rmth1h 1
```

2000 size of the tape that is being used

1 size of the eof mark size

/dev/rmth1h
tape drive device name

0/1 tape controller identification number

/dcelocal*/var/dfs/backup/conf_
The User-Defined Configuration File is always created. If no value is given for **ask, autoquery, name_check** and **file**, then the default value is stored in the User-Defined Configuration File. If **mount** and **unmount** are not defined, they are not stored in that file. If the User-Defined Configuration File already exists for the tape drive device name being configured, then it is overwritten.

mkfilesys.dfs

Purpose

Registers and exports AIX CD-ROM file systems, AIX JFS file systems, or DCE LFS aggregates and filesets into the DFS namespace

Synopsis

mkfilesys.dfs -file_system *file_system_type* [**-aggregate_id** *aggregate_id*]
[**-aggregate_name** *aggregate_name*][**-device_name** *device_name*]
[**-fileset** *fileset_name*][**-mount_point** *DFS_mount_point*][**-root**]

CD-ROM

Exports a CD-ROM system

mkfilesys.dfs -file_system_type cdrom[**-aggregate_id** *aggregate_id*]
-device_name *device_name*
[[**-fileset** *fileset_name*] [**-mount_point** *DFS_mount_point*]]

JFS

Exports a JFS File System containing the Root Fileset (root.dfs)

mkfilesys.dfs -file_system_type jfs [**-aggregate_id** *aggregate_id*]
-device_name *device_name*[-root]

Exports any other JFS File System

mkfilesys.dfs -file_system_type jfs [**-aggregate_id** *aggregate_id*]
-device_name *device_name* **-fileset** *fileset_name*
[**-mount_point** *DFS_mount_point*]

LFS

Exports a new LFS Aggregate without a fileset

mkfilesys.dfs -file_system_type lfs [**aggregate_id** *aggregate_id*]
-aggregate_name *aggregate_name* **-device_name** *device_name*

Creates a root fileset (root.dfs) within an existing LFS aggregate

mkfilesys.dfs -file_system_type lfs **-aggregate_name** *aggregate_name*
-root

Creates a root fileset (root.dfs) within a new aggregate as you export it

mkfilesys.dfs -file_system_type lfs [**-aggregate_id** *aggregate_id*]
-aggregate_name *aggregate_name* **-device_name** *device_name*
-root

Creates any other fileset within an existing LFS aggregate

mkfilesys.dfs -file_system_type lfs
-aggregate_name *aggregate_name*
-fileset *fileset_name* [**-mount_point** *DFS_mount_point*]

Creates any other fileset with a new aggregate as you export it

mkfilesys.dfs -file_system_type lfs[**-aggregate_id** *aggregate_id*]
-aggregate_name *aggregate_name* **-device_name** *device_name*
-fileset *fileset_name* [**-mount_point** *DFS_mount_point*]

OPTIONS

-aggregate_id *aggregate_id*

An integer greater than 0 to be used as the aggregate ID number for the CD-ROM file system, the JFS file system, or the LFS aggregate being exported. It must be unique among all file systems and aggregates on this machine. If you do not specify the -**aggregate_id** option, **mkfilesys.dfs**

uses one more than the highest aggregate ID in the current /opt/dcelocal/var/dfs/dfstab file. Valid values are 1 - 1,000,000.

-aggregate_name *aggregate_name*
The name of the aggregate. If the aggregate is not already exported, **mkfilesys.dfs** attempts to export it. Use only when exporting LFS aggregates.

-device_name *device_name*
The name of the of the block device of the file system or aggregate being exported; for example, /dev/lv02. If LFS, this logical volume must already have been initialized as a DFS LFS file system by running the **newaggr** command.

-file_system_type *file_system_type*
Specifies the type of file system to export. Valid values are cdrom, jfs, and lfs. This option must be specified.

-fileset *fileset_name*
The name of the fileset to be defined or created in this file system or aggregate. The name must be unique within the cell, it must not be more than 102 characters in length, and it must consist of only the following character types:

- alphabetic (a-z, A-Z)
- numeric (0-9)
- period (.)
- dash (-)
- underscore (_)fileset_name

-mount_point *DFS_mount_point*
Specifies the location in the DFS file space where the fileset should be mounted or made available to DFS clients. The default is to not mount the fileset.

The mount point should not exist yet, but the parent directory should. This option cannot be used with the **-root** option or if the DFS client (dfs_cl component) is not configured on this machine.

-root Specifies that the root fileset (root.dfs) should be defined/created in this file system or aggregate. It is automatically mounted at /.:/fs. This is not an option when exporting a CD-ROM file system, when using the -**fileset**, or **-mount_point** options.

DESCRIPTION

CD-ROM

Before running the **mkfilesys.dfs** command, you must add the CD-ROM file system to AIX. If the filesystem is not mounted, **mkfilesys.dfs** mounts it for you. This file system should have `mount=true` in its stanza in the */etc/filesystems* file.

When a CD-ROM file system is exported, a fileset must be defined for it. However, the fileset does not have to have its mount point created when it is defined. For example, you might not want to export the data to DFS clients immediately.

Only one fileset can be defined within a CD-ROM file system.

JFS

Before running the **mkfilesys.dfs** command, you must add the JFS file system to AIX. If the filesystem is not mounted, **mkfilesys.dfs** mounts it for you. This file system should have `mount=true` in its stanza in the */etc/filesystems* file.

When a JFS file system is exported, a fileset must be defined for it. However, the fileset does not have to have its mount point created when it is defined. For example, you might not want to export the data to DFS clients immediately.

Only one fileset can be defined within a JFS partition.

LFS

Before exporting a new LFS aggregate, you must have already created a logical volume and initialized it as a DFS LFS file system by running the **newaggr** command.

When an LFS aggregate is exported, a fileset may or may not be created within it at that time. To create a fileset within an aggregate that has already been exported, use the **-fileset** option, but do not use the **-device_name** option. LFS aggregates can contain more than one fileset.

Common to CD-ROM, JFS, and LFS

To be able to create the mount point for a fileset, the DFS File Server must also be configured as a DFS Client. If you do not use the **-mount_point** option to create the mount point for a fileset when it is defined, you can create the mount point later by issuing the **fts crmount** command from any machine configured as a DFS Client.

A special fileset is *root.dfs*. When it is defined, it is automatically mounted at /.:/fs. You do not need to nor can you use the **-mount_point** option or the **fts crmount** command to create a mount point for the *root.dfs* fileset.

Root.dfs is a valid CD-ROM file system.

The **mkfilesys.dfs** command must be run by the root user. In addition, you must have DCE credentials for a user in the *admin.fl* administration list or for the machine principal (*host/dce_hostname/self*) for the fileset database machine.

To use the **-mount_point** option, you must have *write* and *insert* permission for the parent directory of the mount point.

For information about starting DCE and DFS at system start time, see *IBM DCE for AIX, Version 2.2: Quick Beginnings*.

EXAMPLES

CD-ROM

To export a CD-ROM file system and mount its fileset, making the data available to DFS clients, enter:

```
mkfilesys.dfs -file_system_type cdrom -fileset new.tools
-mount_point /.:/tools/bin -device_name /dev/cd0
```

JFS To export a JFS file system containing the root fileset, enter:

```
mkfilesys.dfs -file_system_type jfs -root -device_name /dev/lv05
```

To export a JFS file system and mount its fileset, making the data available to DFS clients, enter:

```
mkfilesys.dfs -file_system_type jfs -fileset new.tools
-mount_point /.:/fs/tools/bin -device_name /dev/lv06
```

LFS To export an LFS aggregate, creating the root fileset within it, enter:

```
mkfilesys.dfs -file_system_type lfs -root -device_name /dev/lv05
-aggregate_name first.aggregate
```

Notice that */dev/lv05* must already exist and have had the **newaggr** command run on it.

To create a new fileset within an existing aggregate name dept.e94, enter:

```
mkfilesys.dfs -file_system_type lfs -fileset new.tools
-mount_point /.:/fs/tools/bin -aggregate_name dept.e94
```

Because the **-mount_point** option was given, the fileset is mounted so it is immediately available to DFS clients.

To export a new aggregate with the name empty.aggregate, without creating a fileset within it, enter:

```
mkfilesys.dfs -file_system_type lfs -device_name /dev/lv07
-aggregate_id 19 -aggregate_name empty.aggregate
```

The **-aggregate_id** option specifies that aggregate ID 19 should be used to identify this aggregate on this machine.

RELATED INFORMATION

Files:**/opt/dcelocal/var/dfs/dfstab**

Contains information about the LFS aggregates and AIX file systems that can be exported form the machine.

rmbutc.dfs

Purpose

Unconfigures the Backup Tape Controller on a machine and in one command

Synopsis

rmbutc.dfs [-admin_pwd *password*] [**-cell_admin** *cell_admin identification*] [**-tcid** *tcid*]

OPTIONS

-admin_pwd *password*
Specifies the cell administrator password. This is the password for the **cell_admin** on the cell that you wish to unconfigure **butc**. If you do not provide this required parameter on the command line then you receive a prompt for the password.

-cell_admin *cell_admin identification*
The name of the cell administrator's account. This is by default **cell_admin**. If a different cell admin name was used in creation you need to use that name.

-tcid *tcid*
Specifies Tape Controller Identification Number. The value for the **tcid** must be used in the cell you are configuring as well as on the local machine, even if the machine is in more then one cell. By default the value for **tcid** is 0. Valid values range 0 - 1023.

DESCRIPTION

This command can be issued for each Backup Tape Controller in the cell to unconfigure it on a machine in a cell in one command. The bakserver is updated no matter where it is running in the cell. The local file **TapeConfig** is updated and the User-Defined Configuration File is removed, if it has been created previously using **mkbutc.dfs** and no other Backup Tape Controller on the machine uses the same device.

Privilege Required

The issuer must be logged in as **root** on the local machine.

EXAMPLES

Examples

To remove a tape controller 1, enter:

```
rmbutc.dfs -tcid 1
```

To remove a tape controller 5, enter:

```
rmbutc.dfs -tcid 5
```

To remove a tape controller 0, enter:

```
rmbutc.dfs -tcid 0
```

RELATED INFORMATION

Files:

***/dcelocal*/var/dfs/backup/TapeConfig**
Contains information about the tape controllers that are configured on the local system. The format of the file is as follows:

```
2000 1 /dev/rmth0h 0
2000 1 /dev/rmth1h 1
```

2000 size of the tape that is being used

1 size of the eof mark size

/dev/rmth1h
tape drive device name

0/1 tape controller identification number

/dcelocal*/var/dfs/backup/conf_
The files that are removed for each drive device name no longer have an entry in the TapeConfig file.

rmfilesys.dfs

Purpose

Detaches and unregisters AIX CD-ROM file systems, AIX JFS file systems, or LFS aggregates and filesets from the DFS namespace

Synopsis

rmfilesys.dfs [-aggregate_name *aggregate_name*][**-device_name** *device_name*]
[**-force**][**-fileset** *fileset_name*][**-mount_point** *DFS_mount_point*]
[**-override**]

Flags

To detach and unregister a CD-ROM File System:

rmfilesys.dfs[**-device_name** *device_name* | **-fileset** *fileset_name*]
[**-force**] [**mount_point***DFS_mount_point*][**-override**]

To detach and unregister a JFS File System:

rmfilesys.dfs[**- device_name** *device_name* | **-fileset** *fileset_name*]
[**-force**] [**mount_point** *DFS_mount_point*][**-override**]

To delete an LFS Fileset:

rmfilesys.dfs[**-fileset** *fileset_name*[**mount_point** *DFS_mount_point*][**-override**]

To detach an LFS Aggregate:

rmfilesys.dfs-aggregate_name *aggregate_name*[**-force**]

OPTIONS

-aggregate_name*aggregate_name*
Specifies the name of the LFS aggregate to be detached.

-device_name *device_name*
Specifies the CD-ROM or JFS file system (partition) to be detached. Its fileset is also undefined.

-force Forces the CD-ROM of the JFS File System to be detached from DFS even if it is currently in use.

-fileset *fileset_name*
Specifies the name of the fileset to be undefined. If it is a fileset for a CD-ROM or JFS file system, the file system is detached.

-mount_point *DFS_mount_point*
Specifies the name of the mount point to be deleted. The default is not to delete the fileset's mount.

-override
Override confirmation of fileset deletion.

DESCRIPTION

CD-ROM
Refer to the file system by either its device name or the fileset defined for it. In either case, the file system is detached from DFS and the fileset is undefined.

JFS Refer to the file system by either its device name or the fileset defined for it. In either case, the file system is detached from DFS and the fileset is undefined.

LFS When you detach an LFS aggregate, filesets within are not deleted, but become inaccessible to DFS clients.

EXAMPLES

CD-ROM:

To detach a CD-ROM file system from DFS, using its device name, enter:

```
rmfilesys.dfs -mount_point /.:/lfs/tools/bin -device_name /dev/cd0
```

To detach a CD-ROM file system from DFS, using its fileset name, enter:

```
rmfilesys.dfs -fileset new.tools
```

JFS:

To detach a JFS file system from DFS, using its device name, enter:

```
rmfilesys.dfs -mount_point /.:/lfs/tools/bin -device_name /dev/lv05
```

To detach a JFS file system from DFS, using its fileset name, enter:

```
rmfilesys.dfs -fileset new.tools
```

LFS:

To detach an LFS aggregate, by specifying the aggregate name, enter:

```
rmfilesys.dfs -aggregate_name first.aggregate
```

To delete a fileset for an LFS aggregate, enter:

```
rmfilesys.dfs -mount_point /.:/fs/tools/bin -fileset new.tools
```

To delete a fileset for an LFS aggregate, enter:

```
rmfilesys.dfs -mount_point /.:/fs/tools/bin -fileset new.tools
```

RELATED INFORMATION

Files:*/dcelocal*/**var/dfs/dfstab**

The files contain information about the LFS aggregates and AIX file systems that can be exported from the machine.

start.dfs

Purpose

Starts DFS components configured on the local machine.

Synopsis

start.dfs [all][usage] [-?] [help] [operations] *components*

OPTIONS

all Starts the configured DFS components on the local machine.

usage/-?
Displays a help message.

help Displays a brief description for the passed arguments.

operations
Lists all options and components.

components
Specifies the components to be started.

Client Components

all_cl All clients (dfs_cli)

dfs_cl DFS client

Server Components

all_svr
All servers (dfs_svr, dfs_repsvr, dfs_scm, dfs_fldb, dfs_bkdb)

dfs_svr
DFS File Server

dfs_repsvr

dfs_scm
DFS System Control Machine

dfs_fldb
DFS Fileset Database

dfs_bkdb
DFS Backup Database

DESCRIPTION

The **start.dfs** command starts currently configured components on the local machine.

To display configured components use the **show.cfg** command.

Privilege Required

To start DFS components configured on the local machine, the issuer must be logged in as **root** on the local machine. The issuer must be on the **admin** list.

stop.dfs

Purpose

Stops DFS components configured on the local machine.

Synopsis

stop.dfs [all] [usage][-?][help][operations] *components*

OPTIONS

all Stops the DFS components configured on the local machine.

usage/-?
Displays a help message.

help Displays a brief description for the passed arguments.

operations
Lists all options and components.

Client Components

all_cl All clients (dfs_cli)

dfs_cl DFS client

Server Components

all_svr
All servers (dfs_svr, dfs_repsvr, dfs_scm, dfs_fldb, dfs_bkdb)

dfs_svr
DFS File Server

dfs_repsvr

dfs_scm
DFS System Control Machine

dfs_fldb
DFS Fileset Database

dfs_bkdb
DFS Backup Database

DESCRIPTION

The **stop.dfs** command stops currently configured components on the local machine.

To display configured components, use the **show.cfg** command.

Privilege Required

To stop DFS components configured on the local machine, the issuer must be logged in as **root** on the local machine. The issuer must be on the **admin** list.

unconfig.dfs

Purpose

Removes configurations of DCE DFS components

Synopsis

unconfig.dfs [-admin_pwd *password*][**-cell_admin** *cell_admin*]
[**-config_type** *config_type*][**-dce_hostname** *dce_hostname*][**-dependents**][**-force**]
[**-group_rsp_path** *pathname*][**-host_id** *host_id*][**-rsp_file** *filename*]

OPTIONS

-admin_pwd *password*
Specifies the cell administrator password. Caution should be used with this option because of the security risk it poses in making this password accessible to others.

-cell_admin *cell_admin*
The name of the DCE cell administrator's account. If no **-cell_admin** option is given, the account *cell_admin* is assumed.

-config_type (full|local|admin)
Allows the cell administrator to split unconfiguration by specifying admin, local, or full unconfiguration of machines in the cell. The **-config_type** option has three available config_types:

admin Specifies that the admin portion of unconfiguration is completed for the host indicated by the *dce_hostname* flag. This cleans up the CDS namespace and security registry. The user must have cell administrator authority within the cell.

local Specifies that the local portion of unconfiguration is completed for the local machine. This stops the daemons and update or removes appropriate files. The user must have **root** authority on the local machine.

full Specifies full unconfiguration on the local machine. This is the default *config_type*. When doing full unconfiguration on the local host, the user has both cell administrator authority within the cell and *root* authority on the local machine. Full unconfiguration is the equivalent of admin unconfiguration and local unconfiguration combined. If the **-config_type** option is not used, a full unconfiguration is assumed.

-dce_hostname *dce_hostname*
Used with the **-config_type** admin option to identify the host to unconfigure. Use **-dce_hostname** only when doing the admin portion of unconfiguration.

-dependents
Unconfigures dependent components. Specifies that any components that depend on those listed on the command line should also be unconfigured. For example, on a machine with *dfs_srv* and *dfs_repsrv*, **unconfig.dfs -dependents** *dfs_srv* also unconfigures *dfs_repsrv*.

-force Forces unconfiguration of components named on the command line, even if other components depend on their presence. This option should be used in cleanup situations and with caution because a cell can be put into an unstable state.

-group_rsp_path *pathname*
Specifies a directory path to use when searching for included response files.

-host_id *host_id*
Specifies the TCP/IP host name or TCP/IP address of the machine being admin unconfigured. When **unconfig.dfs** is called with **-config_type** admin, the **-host_id** option must also be used. Admin unconfiguration can be used for a machine whose TCP/IP address is not registered with a nameserver. In this case, use the **-dce_hostname** *dce_hostname* option with the **-host_id** *IP_address* option. The **-host_id** option can be used only with the **-config_type** admin option.

-rsp_file *filename*
Specifies the full path name of a response file.

DESCRIPTION

The **unconfig.dfs** command stops the DFS components requested and removes their configuration and database files.

If the DCE cell for which the machine is configured is not available, or you do not have authority to alter the namespace, admin lists, registry database, or Fileset Location Database in that cell, use the **-config_type** *local* option. This limits **unconfig.dfs** to performing only local cleanup operations. The **unconfig.dfs** command also prints a list of the actions the cell administrator needs to take to fully remove the DFS components from the cell itself (for example, removing a Fileset Database machine from the list of those servers in the CDS namespace and security registry).

The kernel daemons **fxd** and **dfsd** can be killed, but you should reboot your machine before attempting to configure these components again.

When a DFS File Server is unconfigured, its CD-ROM file systems, JFS file systems, LFS aggregates, and filesets are not altered. To remove these entities from the Fileset Location Database, use the **rmfilesys.dfs** command.

If the user attempts to remove the file server without also removing the replicated file server (repserver), the **unconfig.dfs** command produces an error message. The replicated file server is dependent on the DFS file server. In other words, the repserver cannot run alone. Use the **-force** option to overcome this dependency. A user using the **-dependents** option can override dependency checking and forcefully remove a file server without also removing the repserver. Or the user can specify the **-dependents** option and the **dfs_srv** component, and **unconfig.dfs** component automatically removes the **dfs_repsrv** component because it is dependent on the presence of **dfs_srv**.

If you do not use **-config_type** *local*, you are prompted for the password for the DCE cell administrator's account so **unconfig.dfs** can perform unconfiguration tasks that require DCE authentication.

If the environment variable **cell_admin_pw** is set, **unconfig.dfs** uses its value for the cell administrator's password without prompting you. This feature can be useful when automating unconfiguration tasks, but should be used sparingly because of the security risk it poses by making this password accessible to others.

EXAMPLES

To remove all DFS configuration from a machine when the cell administrator's account name is *ca*, enter:

```
unconfig.dfs -cell_admin ca all
```

To remove all DFS server configurations from a machine on which DCE communications are not functioning properly, enter:

```
unconfig.dfs -config_type local all_srv
```

The **-config_type** *local* option prevents **unconfig.dfs** from attempting to remove information from the cell's namespace, registry, administration lists, and fileset location database.

To unconfigure both the **dfs_srv** and **dfs_repsrv** components, assuming both the file server and repserver are running, enter:

```
unconfig.dfs dfs_srv dfs_repsrv
```

or

```
unconfig.dfs -dependents dfs_srv
```

The second example above unconfigures both components because the **-dependents** option specifies that all components dependent on **dfs_srv** are unconfigured.

To forcefully remove the **dfs_srv** component, even though the **dfs_repsrv** component is configured (assuming both the file server and the repserver are running), enter:

```
unconfig.dfs -force dfs_srv
```

This leaves the **dfs_repsrv** component running alone. The user is not warned of any consequences.

If the user attempts to use `unconfig.dfs dfs_srv` when the **dfs_repsrv** component is configured, the **unconfig.dfs** fails and an error message results.

RELATED INFORMATION

- Commands: **config.dfs**, **mkfilesys.dfs**, **rmfilesys.dfs**, **start.dfs**, **stop.dfs**
- Components:

  all All DFS components

  all_cl same as dfs_cl

  all_srv
  All DFS servers (**dfs_bkdb**, **dfs_fldb**, **dfs_repsrv**, **dfs_scm**, **dfs_srv**)

  dfs_bkdb
  DFS backup database machine

  dfs_cl DFS client machine

  dfs_fldb
  DFS fileset database machine

  dfs_repsrv
  DFS replicated file server

dfs_scm
DFS system control machine

dfs_srv
DFS file machine

Chapter 14. Administrative Commands

dfs_intro

Purpose

Introduction to the DFS commands

DESCRIPTION

Most DFS commands are divided into the following categories, or command suites:

bak Operates the DFS Backup System

bos Operates the Basic OverSeer (BOS) Server

cm Configures the Cache Manager

dfstrace
Provides DFS kernel and server process logging information

fts Manipulates filesets

In addition, DFS provides a number of miscellaneous commands (for example, **salvage** and **scout**) not associated with a specific command suite. DFS also provides an additional command, **dfsiauth**, that is used with the NFS/DFS Authenticating Gateway.

System administrators use the majority of DFS commands. However, DCE users can use the following commands:

- The **cm** commands **cm_statservers** and **cm_whereis** to determine machine, file, and directory information
- The **fts** command **fts_lsquota** to check quota information

DFS Command Types

DFS commands follow these general naming rules. Commands that begin with

- **add** or **rm** (remove) affect lists or groups of DFS objects. For example, **bos addadmin** adds an administrative user to an administrative list.
- **cr** (create) or **del** (delete) affect DFS objects. For example, **fts crserverentry** creates a DFS object, a server entry.
- **ls** (list) are used to display objects and groups of objects.
- **set** are used to assign values to parameters; for example, **fts setrepinfo** assigns replication parameters. Analogously, commands beginning with **get** are used to display parameters; for example, **cm getcachesize** displays parameters used by the Cache Manager.

Rules For Using DFS Commands

When supplying an argument to a command, the option associated with the argument can be omitted if

- All arguments supplied with the command are entered in the order in which they appear in the command's synopsis.
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.
- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

In the case where two options are presented in { | } (braces separated by a vertical bar), the option associated with the first argument can be omitted if that argument is provided; however, the option associated with the second argument is required if that argument is provided.

If it must be specified, an option can be abbreviated to the shortest possible form that distinguishes it from other options of the command. For example, the **-server** option found in many DFS commands can typically be omitted or abbreviated to be simply **-s**.

It is also valid to abbreviate a command name to the shortest form that still distinguishes it from the other command names in the suite. For example, it is acceptable to shorten the **bos install** command to **bos i** because no other command names in the **bos** command suite begin with the letter ″i.″ However, there are three **bos** commands that begin with the letter ″g″: **bos getdates**, **bos getlog**, and **bos getrestart**. To remain unambiguous, they can be abbreviated to **bos getd**, **bos getl**, and **bos getr**.

The following examples illustrate three acceptable ways to enter the same **bos getlog** command.

Complete command:

```
$ bos getlog -server /.../abc.com/hosts/fs1 -file BosLog
```

Abbreviated command name and abbreviated options:

```
$ bos getl -s /.../abc.com/hosts/fs1 -f  BosLog
```

Abbreviated command name and omitted options:

```
$ bos getl /.../abc.com/hosts/fs1 BosLog
```

Aliases

An alias is an alternative way of entering an existing command. Each alias is either shorter than the original command, or it is unique within the command's suite. (Because only the number of characters sufficient to uniquely identify a command need to be entered to execute the command, unique aliases require less typing.)

The **bak** suite is the only command suite with aliases. Refer to the **bak(8dfs)** reference page for a list of the **bak** commands that have aliases.

Receiving Help

There are several different ways to receive help about DFS commands. The following list summarizes the syntax for the different help options:

Reference pages for a command suite

To view the introductory page for a command suite, enter **man** followed by the command suite:

```
$ man bak
```

Reference page for an individual command

To view the reference page for a command in a suite, enter **man** followed by the command suite and the command name. Use an _ (underscore) to connect the command suite to the command name. *Do not use the underscore when issuing the command in DFS.*

```
$ man bak_command
```

List of commands in a command suite

To view a list of all commands in a command suite, enter the command suite name followed by **help**:

```
$ bak help
```

The command syntax for a single command

To view the syntax of a specific command, enter the suite name, **help**, and the command name, in that order:

```
$ bak help command
```

In addition, all DFS commands include a **-help** option you can use to display the syntax of the command.

The **apropos** command displays the first line of the online help entry for any command that has a specified string in its name or short description; this is useful if you cannot remember the exact name of a command. If the string is more than a single word, surround it with " " (double quotes) or other delimiters; enter all strings in lowercase letters. For example, the following command produces a list of all **bos** commands with the word **create** in their names or short descriptions:

```
$ bos apropos -topic create
```

Privileges Required

The majority of DFS commands, because they are administrative in nature, require that the issuer be included in an **admin** file (for example, **admin.bos**). Some commands require that the issuer have specific permissions to access files (for example, the delete permission on a directory) or be logged in as **root** on the machine on which the command is issued. The exact privilege needed to execute each command is detailed with the command.

CAUTIONS

Specific cautionary information is included with individual commands.

RELATED INFORMATION

For more information about the commands in a specific suite and a list of the commands in the suite, see the introductory page for that suite.

bak(8dfs)

bos(8dfs)

cm(8dfs)

dfstrace(8dfs)

fts(8dfs)

bak

Purpose

Introduction to the **bak** command suite

OPTIONS

The following options are used with many **bak** commands; they are also listed with the commands that use them:

-server *machine*
Specifies the File Server machine to use with the command. You can use any of the following to specify the File Server machine:

1. The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*)
2. The machine's host name (for example, **fs1.abc.com** or **fs1**)
3. The machine's IP address (for example, **11.22.33.44**)

-tapehost *machine*
Specifies the machine for which a Tape Coordinator is being added. You can use the machine's DCE pathname, its host name, or its IP address.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator being used to execute the command. Legal values for this argument are the integers 0 (zero) to 1023. Because the default for the TCID is **0** (zero), the drive used most often should be assigned a TCID of **0** (zero).

-help Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see the **dfs_intro(8dfs)** reference page.

DESCRIPTION

Commands in the **bak** command suite are issued by system administrators to work with the DFS Backup System. The commands copy user and system files to backup tapes and restore information from the tapes, if necessary. All **bak** commands are restricted to administrative users only.

The Backup System relies primarily on the following two types of machines and the information and services they provide:

- *Backup Database machines*, which are server machines that house the DFS Backup Database. A cell must have at least one Backup Database machine to use the Backup System; it is recommended that a cell have at least three Backup Database machines. The Backup Database stores two types of records: dump set records, which list the fileset families and tapes in the dump sets; and administrative records, which list fileset families and their entries, as well as dump levels and tape hosts.
- *Tape Coordinator machines*, which are client or server machines with attached tape drives. A Tape Coordinator machine runs an instance of the **butc** process, which is the Backup Tape Coordinator process, for each drive. A Tape Coordinator process controls the behavior of its associated drive and accepts service requests from the Backup System.

  A Tape Coordinator ID (TCID) acts as an identifier for the Tape Coordinator. The TCID for each Tape Coordinator is assigned in the **TapeConfig** file on the

machine that houses the tape drive and in the Backup Database. Each TCID is unique to the cell in which the Tape Coordinator is used. With **bak** commands, the TCID specifies the Tape Coordinator to use with the command.

Interactive Mode

The **bak** command suite can be used in regular command mode or in interactive mode. To enter interactive mode, enter **bak** at a command shell prompt. While you are using this mode, the following information applies:

- The word **bak** does not need to be entered with each command; the **bak>** prompt replaces the command shell prompt.
- Regular expression characters do not need to be escaped; in regular command mode, all regular expression characters must be placed in ″ ″ (double quotes) or escaped with a **** (backslash).
- New connections do not have to be established to the **bakserver** and **flserver** processes, as necessary, each time a command is issued, so execution time is faster than in noninteractive mode.
- Multiple operations can be tracked with the **bak jobs** command; in regular command mode, pending operations cannot be tracked.
- Currently executing and pending operations can be canceled with the **bak kill** command; in regular command mode, the **bak kill** command cannot be used.

Descriptions of the **bak jobs**, **bak kill**, and **bak quit** interactive commands follow; interactive commands can be issued *only* in interactive mode (at the **bak>** interactive prompt).

The bak jobs Command

The **bak jobs** command lists the job ID number the Backup System has assigned to each dump and restore operation for a Tape Coordinator; the listed operations can be currently executing or pending. The job ID number is not the same as the unique dump ID number assigned to each dump set by the Backup System. (It is also not the same as the task ID number that is sometimes displayed in the output of certain commands; the task ID number can always be safely ignored.)

The complete syntax for the command is

```
jobs [-help ]
```

The **-help** option displays the online help for the command.

If no operations are executing or pending, the **bak>** prompt returns immediately. Otherwise, the output includes one line for each operation, reporting

- The job ID number.
- A name describing the operation.
- The number of kilobytes transferred so far (from file system to tape for a dump operation, from tape to file system for a restore operation).
- For a dump operation, the string **fileset** followed by the name of the fileset currently being dumped; for a restore operation, the string **fileset** followed by the name of the fileset currently being restored.
- A message indicating the status of the operation. No message is displayed if the operation is executing normally.

The bak kill Command

The **bak kill** command terminates a currently running or pending dump, restore, or tape labeling operation. If the command interrupts a backup operation, all filesets written to the tape before the kill signal is received are complete and usable. The fileset being written when the signal is received may not be complete and *should not be used*. It is best not to use any of the filesets from an interrupted dump.

If the command interrupts a restore operation, all completely restored filesets are online and usable. Because complete restoration of a fileset usually requires data from multiple tapes (a full dump tape and one or more incremental dump tapes), most filesets are usually not completely restored. If the kill signal occurs before the system accesses all of the necessary tapes, most filesets are not restored to the desired date or version and *should not be used*.

If the interrupted restore is overwriting one or more existing filesets, the filesets can be lost entirely; however, the data being restored still exists on tape. In general, to avoid the inconsistencies that can result from an interrupted restore operation, reinitiate the restore operation.

The complete syntax for the command is

```
kill -job  {jobID | dump_set} [-help ]
```

The **-job** option identifies the operation to kill. It can be

- The job ID of the operation, as displayed in the output of the **bak jobs** command.
- The name of the operation, as displayed in the output of the **bak jobs** command if the operation is a dump. Dump set names associated with dump operations have the form *fileset_family_name***.** *dump_level*. It is not possible to distinguish restore operations by name.

The **-help** option displays the online help for the command. All other valid options specified with the **-help** option are ignored.

The bak quit Command

The **bak quit** command exits interactive mode; the regular shell prompt replaces the **bak>** prompt.

The complete syntax for the command is

```
quit [-help ]
```

The **-help** option displays the online help for the command.

Command and Monitoring Windows

When using the Backup System, you can use a single terminal session as the command window in which to issue **bak** commands to the Tape Coordinators on all Tape Coordinator machines. In addition, you must open a separate monitoring session for each Tape Coordinator process running on a Tape Coordinator machine. The Tape Coordinator process runs in the foreground; any prompts from the Backup System appear in this window.

Aliases

An alias is an alternate way of entering a command. Each alias is either shorter than the original command or it is unique within the command's suite. (Because only the number of characters sufficient to uniquely identify a command need to be entered to execute the command, unique aliases require less typing.)

The **bak** suite is the only command suite with aliases. The following commands in the **bak** suite can also be entered as specified:

bak restoredb
Can be entered as **bak dbrestore**.

bak restoredisk
Can be entered as **bak dkrestore**.

bak restoreft
Can be entered as **bak ftrestore**.

bak restoreftfamily
Can be entered as **bak familyrestore**.

Cautions

Specific cautionary information is included with individual commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ man bak
Displays the reference page for the command suite.

$ man bak_ *command*
Displays the reference page for an individual command. You must use an _ (underscore) to connect the command suite to the command name. *Do not use the underscore when issuing the command in DFS.*

$ bak help
Displays a list of commands in a command suite.

$ bak help *command*
Displays the syntax for a single command.

$ bak apropos -topic *string*
Displays a short description of any commands that match the specified *string*.

Consult the **dfs_intro(8dfs)** reference page for complete information about the DFS help facilities.

Privilege Required

It is recommended that all system administrators using the Backup System be included in the following lists: the **admin.bak** file on all machines that house the Backup Database, the **admin.fl** file on all machines that house the Fileset Location Database (FLDB), and the **admin.ft** file on all File Server machines. The issuer of a **bak** command must be included in the **admin.bak** list on all machines that house the Backup Database.

RELATED INFORMATION

Commands: **bak adddump(8dfs)**, **bak addftentry(8dfs)**, **bak addftfamily(8dfs)**, **bak addhost(8dfs)**, **bak apropos(8dfs)**, **bak deletedump(8dfs)**, **bak dump(8dfs)**, **bak dumpinfo(8dfs)**, **bak ftinfo(8dfs)**, **bak help(8dfs)**, **bak labeltape(8dfs)**, **bak lsdumps(8dfs)**, **bak lsftfamilies(8dfs)**, **bak lshosts(8dfs)**, **bak readlabel(8dfs)**, **bak restoredb(8dfs)**,

bak restoredisk(8dfs), **bak restoreft(8dfs)**, **bak restoreftfamily(8dfs)**, **bak rmdump(8dfs)**, **bak rmftentry(8dfs)**, **bak rmftfamily(8dfs)**, **bak rmhost(8dfs)**, **bak savedb(8dfs)**, **bak scantape(8dfs)**, **bak setexp(8dfs)**, **bak status(8dfs)**, **bak verifydb(8dfs)**, **dfs_intro(8dfs)**.

bak adddump

Purpose

Defines a dump level in the dump hierarchy

Synopsis

bak adddump -level *dump_level...*[**-expires** *date*][**-help**]

OPTIONS

-level *dump_level*

Names each new dump level to be added to the dump hierarchy. Specify a full pathname for each dump level. Precede the name of each level by a / (slash); the / (slash) is a metacharacter that separates each level in a dump level name. When defining a full dump level, precede the name of the level with a / (slash). When defining an incremental dump level, precede the name of each dump level in the name with a / (slash); the elements in the pathname preceding the last one must already exist in the dump hierarchy. The complete pathname of each dump level must be unique within the Backup Database of the local cell.

Dump level names can have any number of elements. Each element cannot contain more than 28 characters. Complete dump level names cannot contain more than 255 characters. They can include any characters. (To avoid confusion when dump set names are created, the name should not include a dot. When a dump set is transferred to tape, the fileset family name and the last component of the dump level name are joined by a dot to form the name of the dump set.) When including regular expression characters, escape each character with a **** (backslash) or " " (double quotes).

-expires *date*

Defines the expiration date to be associated with each new dump level. Expiration dates can be specified as absolute or relative values. Absolute expiration dates have the format

at *mm*/*dd*/*yy* [*hh*:*mm*]

The word **at** is followed by a date (*month*/ *day*/ *year*) and, optionally, a time (*hours*:*minutes*). When the system creates a dump set at this level, it assigns the specified date as the expiration date of the tape that contains the dump set.

Valid values for *yy* are 00 to 37, which are interpreted as the years 2000−2037, and 70 to 99, which are interpreted as 1970−1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038−2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038.

If specified, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The default time is **00:00** (12:00 a.m.).

Relative expiration dates have the format

in [*integer***y**] [*integer***m**] [*integer***d**]

The word **in** is followed by a number of years (maximum 9999), months (maximum 11), and days (maximum 30), or a combination of these arguments. When the system creates a dump set at this level, it adds the specified values to the current date to calculate the expiration date of the tape that contains the dump set. At least one of the three values must be specified, and the appropriate unit abbreviation (**y**, **m**, or **d**) must always accompany a value. If more than one of the three is specified, they must appear in the order shown. As with absolute dates, a number of years that causes the relative time to exceed the year 2038 is effectively truncated to the number of years remaining until 2038.

If you omit this option, tapes created at the specified dump levels have no expiration dates, meaning they can be overwritten by appropriately named dump sets at any time. Although the **-expires** option is followed by an ellipsis, you can specify only one expiration date. (The ellipsis is included to accommodate the DFS command parser.)

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak adddump** command defines one or more dump levels in the dump hierarchy that is stored in the Backup Database and names them as specified by **-level**. Precede each different level in a dump level name with a **/** (slash) metacharacter. If a dump level is for full dumps, provide only its name preceded by a **/** (slash) (for example, **/full**).

If a dump level is for incremental dumps, its name resembles a pathname listing the dump levels that serve as its parents, starting with a full dump level and proceeding (in order) down the hierarchy. The dump level's immediate parent (named by the next-to-last element in the pathname) is the reference point that determines which files are included in dump sets made at the dump level. Files with modification time stamps later than the date and time when the volume was dumped at the parent dump level are included.

The optional **-expires** option associates an expiration date with each dump level. The expiration date is applied to tapes containing dump sets made at the dump level; after the specified date, the Backup System overwrites the tape's contents with acceptably named dump sets without question.

An attempt to overwrite an unexpired tape fails until the issuer relabels the tape with the **bak labeltape** command. (Because the label records the unexpired expiration date or unacceptable name, erasing the label removes the obstacle to overwriting.) If no expiration date is defined for a tape, the Backup System overwrites the dump set on the tape with a dump set of the same name without question. Expiration dates can be either absolute or relative; see the Options section for details.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command defines a full dump called **/yearly** with a relative expiration date of one year:

```
$ bak addd -level /yearly -expires in 1y
```

The following command defines an incremental dump called **/full/incr1** with a relative expiration date of 3 months and 15 days:

```
$ bak addd -l /full/incr1 -e in 3m 15d
```

The following command defines two dump levels, **week1** and **week2**; both are incremental from the parent, **monthly**, and both are defined to expire at 12:00 a.m. on 1 January 1992:

```
$ bak adddump -l /monthly/week1 /monthly/week2 -e at 01/01/92
```

RELATED INFORMATION

Commands: **bak dump(8dfs)**, **bak labeltape(8dfs)**, **bak lsdumps(8dfs)**, **bak rmdump(8dfs)**.

bak addftentry

Purpose

Defines a fileset family entry in a fileset family

Synopsis

bak addftentry -family *fileset_family_name* **-server** *machine* **-aggregate** *name* **-fileset** *name*[**-help**]

OPTIONS

-family *fileset_family_name*
Names the fileset family to which this fileset family entry is to be added. The fileset family must already have been created with the **bak addftfamily** command.

-server *machine*
Indicates the File Server machines that house the filesets in the fileset family entry. Legal values for a single machine are the machine's DCE pathname, the machine's host name, or the machine's IP address. You can also specify the regular wildcard expression (**.***) to match all machine names; in noninteractive mode, surround the wildcard with double quotes (″**.***″).

-aggregate *name*
Indicates the aggregates that house the filesets in the fileset family entry. Legal values are the device name or aggregate name of an aggregate (these names are specified in the first and second fields of the entry for the aggregate in the *dcelocal***/var/dfs/dfstab** file) or the regular wildcard expression (**.***), which matches any aggregate name. In noninteractive mode, surround the wildcard with double quotes (″**.***″).

-fileset *name*
Indicates the filesets to be included in the fileset family entry. Legal values are a specific fileset name, the regular wildcard expression (**.***), or a regular expression that includes the regular expression characters described in the **Description** section of this reference page. In noninteractive mode, surround the entire argument with ″ ″ (double quotes) if it contains regular expression characters, or escape each regular expression character with a **** (backslash); otherwise, the command shell attempts to interpret the characters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak addftentry** command adds a fileset family entry to the fileset family specified with the **-family** option. The fileset family must already have been created with the **bak addftfamily** command.

A fileset family entry can include different numbers and groupings of filesets, depending on how the **-server**, **-aggregate**, and **-fileset** options are combined. For the **-server** and **-aggregate** options, the issuer can specify either a single, specific value or the wildcard (**.***). The wildcard matches any string, so it matches every

server machine name or aggregate name found in the Fileset Location Database (FLDB). The **bak** program initiates a search of the entire FLDB to resolve the wildcards.

For the **-fileset** option, a wider range of notation from the regular expression character set is acceptable and can be combined with specific character strings. Regular expression characters are case sensitive. In addition to strings of individual letters (which match any occurrence of that exact string) and the wildcard (**.***, which matches any fileset name), the acceptable notation includes the following regular expression characters. Note that these characters cannot be used for server machine or aggregate names.

*** (asterisk)**
Matches any number of repetitions (0 or more) of the previous character and can be combined with any other regular expression character.

[ ] (brackets)
Around a list of characters, matches a single instance of any of the characters, but no other characters. For example, **[abc]** matches **a** or **b** or **c** but not **d** or **A** or **ab**.

^ (caret)
When used as the first character in a bracketed set, indicates a match with any single character except the characters that follow it. For example, **[^a]** matches any single character except lowercase **a**.

? (question mark)
Matches any single character or no character. For example, **?** matches **a** or **A** or **1** or *****.

. (dot) Matches any single character, but a character must be present.

\ (backslash)
Can precede any of the regular expression characters in this list so that they match only their literal values. For example, the expression ***** matches a single asterisk, and the expression **** matches a single backslash.

In the following example, the combination of letters and regular expression characters matches any string that begins with a **user.** prefix and ends with a **.bak** extension:

user\..*\.bak

The previous example is issued in interactive mode. When issuing this command in noninteractive mode, it is necessary to enclose character strings that include regular expression characters in ″ ″ (double quotes) or to escape the regular expression characters with the **** (backslash); for example, ″**user\..*\.bak**″ and **user\\..*\\.bak** are equivalent to the previous example. Otherwise, the command shell attempts to resolve the regular expression characters rather than pass them to the **bak** command interpreter for resolution. This can result in failure of the command or creation of incorrect fileset entries.

Possible values for the arguments of the **bak addftentry** command follow. To create a fileset family entry that includes

- Every fileset in the cell's file system, provide the **.*** wildcard for all three options.
- Every fileset on a machine, provide the DCE pathname of the machine with -server and the **.*** wildcard for **-aggregate** and **-fileset**.

- Every fileset on every aggregate of the same name, provide the aggregate name with **-aggregate** and the **.*** wildcard for **-server** and **-fileset**.
- Every fileset in the cell's file system that includes a common string of letters in its name (such as a **.bak** extension), provide the **.*** wildcard for **-server** and **-aggregate** and a character string/regular expression combination for **-fileset**.
- Every fileset on one aggregate, provide the DCE pathname of the machine with **-server**, the aggregate name with **-aggregate**, and the **.*** wildcard for **-fileset**.
- Every fileset on a specific machine that includes a common string of letters in its name (such as a **.bak** extension), provide the DCE pathname of the machine with **-server**, the **.*** wildcard for **-aggregate**, and a character string/regular expression combination for **-fileset**.
- Every fileset on each machine's similarly named aggregate that includes a common string of letters in its name (such as a **.bak** extension), provide the **.*** wildcard for **-server**, the aggregate name for **-aggregate**, and a character string/regular expression combination for **-fileset**.
- Every fileset on one aggregate that includes a common string of letters in its name (such as a **.bak** extension), provide the DCE pathname of the machine with **-server**, the aggregate name with **-aggregate**, and a character string/regular expression combination for **-fileset**.
- A single fileset, provide the DCE pathname of the machine with **-server**, the aggregate name with **-aggregate**, and the fileset name with **-fileset**.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following commands add a fileset family entry that includes all filesets in the cell that begin with a **user.** prefix to the fileset family called **user**. The two commands, issued in noninteractive mode, are equivalent.

```
$ bak addftentry user ".*" ".*" "user\..*"
$ bak addftentry user ".*" ".*"user\\..\*
```

Both of the previous commands could be issued in interactive mode as

```
bak> addftentry user.*.* user\..*
```

RELATED INFORMATION

Commands: **bak addftfamily(8dfs)**, **bak lsftfamilies(8dfs)**, **bak rmftentry(8dfs)**.

Files: **dfstab(4dfs)**.

bak addftfamily

Purpose

Creates a new (empty) fileset family in the Backup Database

Synopsis

bak addftfamily -family *fileset_family_name* [**-help**]

OPTIONS

-family *fileset_family_name*
Names the new fileset family. The fileset family name must be unique within the Backup Database of the local cell. It can be no longer than 31 characters, and it can include any characters. (To avoid confusion when dump set names are created, the name should not include a dot. When a dump set is transferred to tape, the fileset family name and the last component of the dump level name are joined by a dot to form the name of the dump set.)

Regular expression characters used in the name of the fileset family must be escaped with a **** (backslash) to prevent the command shell from expanding them when working in noninteractive mode; for example, **usr*** for a fileset family named **usr***. Because they have no meaning in the name of a fileset family, regular expression characters are not recommended.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak addftfamily** command creates a new fileset family in the Backup Database, assigning it the name specified with the **-family** option. To make it easier to track its contents, the fileset family name should give some indication of the filesets it contains (for example, **user** for the fileset family that includes all user filesets in the file system).

Do not include dots in the fileset family name. The names of tapes that contain dump sets of this fileset family consist of the fileset family name and the final component of the dump level name joined by a dot.

After issuing this command, enter the **bak addftentry** command to define the fileset entries included in the fileset family. Use the **bak lsftfamilies** command to list the fileset families currently defined in the Backup Database. Use the **bak rmftfamily** command to remove a currently defined fileset family from the Backup Database.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command creates a fileset family called **sys**:

```
$ bak addftf sys
```

RELATED INFORMATION

Commands: **bak addftentry(8dfs)**, **bak lsftfamilies(8dfs)**, **bak rmftfamily(8dfs)**.

bak addhost

Purpose

Adds a Tape Coordinator entry to the Backup Database

Synopsis

bak addhost -tapehost *machine*[**-tcid** *tc_number*][**-help**]

OPTIONS

-tapehost *machine*
: Names the machine for which the Tape Coordinator is to be added. You can specify the machine's DCE pathname (for example, */.../abc.com/hosts/bak1*), the machine's host name (for example, **bak1.abc.com**), or its IP address (for example, **11.22.33.44**).

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) to be assigned to the Tape Coordinator. Legal values are integers from 0 to 1023. A value must match the TCID assigned to the Tape Coordinator in the */dce/local/***var/dfs/backup/TapeConfig** file on the **-tapehost** machine, and it must be unique among TCIDs in the Backup Database of the local cell. Each Tape Coordinator must have its own TCID, but the TCIDs need not be assigned in sequence (for example, it is legal to skip numbers or to assign them out of order). If this option is omitted, a value of **0** (zero) is used.

  Issuing **bak** commands is most convenient if the Tape Coordinator used most often has a TCID of **0** (zero). The **-tcid** option can then be omitted to direct commands to that Tape Coordinator.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak addhost** command creates an entry for a Tape Coordinator in the Backup Database. The entry indicates

- The machine for which the Tape Coordinator is defined (specified by **-tapehost**).
- The Tape Coordinator's TCID (specified by **-tcid**).
- The UUID of the tape coordinator (generated automatically when the command is issued). The UUID is used by the **bak** commands to identify the tape coordinator that is to perform an operation.

Repeat the command once for each Tape Coordinator on a Tape Coordinator machine. The Backup Database allows a maximum of 1024 Tape Coordinators in the local cell.

The mapping between the TCID of a Tape Coordinator and the device name of the drive with which it is associated is recorded in the **TapeConfig** file on the Tape Coordinator machine. The **TapeConfig** file must be altered accordingly when this command is issued.

Enter the **bak lshosts** command to list the Tape Coordinators that have entries in the Backup Database. Enter the **bak rmhost** command to remove the entry for a Tape Coordinator from the Backup Database.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command creates an entry in the Backup Database for a Tape Coordinator on the machine named **bak1**. The Tape Coordinator is assigned a TCID of **0** (zero); the mapping between the TCID of the Tape Coordinator and the device name of a tape drive must appear in the **TapeConfig** file.

```
$ bak addhost /.../abc.com/hosts/bak1
```

The following command creates an entry in the Backup Database for a Tape Coordinator on the machine named **bak2**; the Tape Coordinator has a TCID of **1**.

```
$ bak addh /.../abc.com/hosts/bak2 1
```

RELATED INFORMATION

Commands: **bak lshosts(8dfs)**, **bak rmhost(8dfs)**.

Files: **TapeConfig(4dfs)**.

bak apropos

Purpose

Shows each help entry containing a specified string

Synopsis

bak apropos -topic *string*[**-help**]

OPTIONS

-topic *string*
Specifies the keyword string for which to search. If it is more than a single word, surround the string with " " (double quotes) or other delimiters. Type all strings for **bak** commands in lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak apropos** command displays the first line of the help entry for any **bak** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **bak help** command.

Privilege Required

No privileges are required.

OUTPUT

The first line of the online help entry for a command lists the command and briefly describes its function. This command shows the first line for any **bak** command where the string specified by **-topic** is part of the command name or first line.

EXAMPLES

The following command lists all **bak** commands containing the word **tape** in their names or short descriptions:

```
$ bak ap tape
labeltape: label tape
readlabel: read label on tape
scantape: list filesets on tape
status: get tape coordinator status
```

RELATED INFORMATION

Commands: **bak help(8dfs)**.

bak deletedump

Purpose

Deletes the record of a dump set from the Backup Database

Synopsis

bak deletedump -id *dumpID*[**-help**]

OPTIONS

-id *dumpID*
The dump ID number of the dump set to be deleted from the Backup Database. Use the **bak dumpinfo** command to list the current dump IDs from the Backup Database.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak deletedump** command removes the record of the dump set associated with the specified dump ID from the Backup Database. It can be used to remove from the Backup Database the record of a dump that contains incorrect data or for which the corresponding tape is to be discarded.

After the record of a dump set is deleted from the Backup Database, dump sets for which it serves as the parent, either directly or indirectly, can no longer be used to restore data to the file system. The **bak deletedump** command can be reissued to remove the record of such dumps from the Backup Database, but leaving a record of them in the database causes no problems. Also, as long as the tape that contains the parent dump set remains available, the **bak scantape** command can be used to restore information about that dump set from the tape to the Backup Database, again making the dump sets that rely on the parent dump set usable.

Use the **bak dumpinfo** command to list the dump IDs currently recorded in the Backup Database.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command deletes the record of the dump set with dump ID **653777462** from the Backup Database:

```
$ bak del 653777462
```

RELATED INFORMATION

Commands: **bak dump(8dfs)**, **bak dumpinfo(8dfs)**, **bak scantape(8dfs)**.

bak dump

Purpose

Dumps a specific fileset family at a specific dump level

Synopsis

bak dump -family *fileset_family_name***-level** *dump_level*[**-tcid** *tc_number*]
[**-noaction**][**-help**]

OPTIONS

-family *fileset_family_name*
Names the fileset family (already defined in the Backup Database using the **bak addftfamily** and **bak addftentry** commands) to be dumped.

-level *dump_level*
Indicates the dump level (already defined in the Backup Database using the **bak adddump** command) to be used in dumping the fileset family. Provide a full pathname for the dump level, including all necessary / (slashes). This option determines whether the dump is full or incremental and, in the latter case, determines which dump level serves as the parent for the dump.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive containing the tape. If omitted, it defaults to **0** (zero).

-noaction
Displays all filesets that would be included in the indicated dump without actually performing the dump. This lets you check a fileset family's size before actually dumping it so that you can calculate the correct number of tapes needed. Specify all other options as you would to actually perform the operation.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak dump** command dumps the fileset family specified by **-family** at the dump level specified as a pathname by **-level** . There are two types of dumps:

- A **full dump** records the structure of all directories in each fileset in the fileset family and includes all the data in each fileset.
- An **incremental dump** also records the structure of all directories in each fileset in the fileset family, but it includes data from only those files in the filesets that changed since the fileset family was dumped at the parent dump level; such files have modification time stamps later than the date and time at which the fileset family was dumped at the parent dump level. The program uses the next-to-last element in the **-level** pathname as the parent dump and consults the Backup Database to learn the date and time at which this fileset family was last dumped at that level.

  If the program cannot locate a dump set dumped at a parent dump level, it looks recursively in the Backup Database for a dump set created at the dump level one higher in the pathname. If it can find no dump set created at a higher dump level in the hierarchy, it creates a full dump set.

If the Backup System is unable to access a fileset (for example, because of a File Server machine or Fileset Server outage), it attempts to access the fileset three times over the course of the operation. If it cannot access the fileset after the third attempt, it omits the fileset from the dump instead of stopping the dump entirely. If the Tape Coordinator performing the dump was initialized at debug level 1, a report on the failure to include the fileset appears in the Tape Coordinator's monitoring window. The Tape Coordinator's error file also records the fileset's omission.

If the failure to access a fileset occurs during a full dump, the next incremental dump of the fileset includes the entire fileset. If the failure occurs during an incremental dump, the next incremental dump of the fileset includes all files modified since the fileset was last included in a dump set.

Before writing the dump to tape, the Tape Coordinator checks that the tape in the indicated tape drive has an acceptable name on its label. If the name on the label is not acceptable, the Backup System prompts for the correct tape. There are three acceptable types of names:

- The tape is labeled *fileset_family_name.dump_level.index*, where *fileset_family_name* and *dump_level* match the values provided on the command line (with **-family** and **-level**). The *dump_level* is the last component of the specified dump level; the *index* distinguishes this tape from others that contain this same dump set. If a single tape contains the entire dump set, its index is 1.
- The tape is labeled as empty. The Backup System labels the tape with the correct name of the form *fileset_family_name.dump_level.index*.
- The tape is not labeled because it has never been used in the Backup System. The Backup System labels the tape with the correct name of the form *fileset_family_name.dump_level.index*.

If it finds that the name on the tape label is acceptable, the Backup System checks the expiration date on the tape before it writes data to it. If the expiration date has not expired, the system does not write data to the tape unless the issuer relabels the tape with the **bak labeltape** command (because the label records the expiration date, erasing the label removes the obstacle to overwriting). If the expiration date has expired or if no expiration date is associated with the tape, the system overwrites the contents of the tape without question (given that the tape has an acceptable name).

The tape label also tells the Tape Coordinator the size of the tape. However, the Tape Coordinator applies the capacity specified in the **TapeConfig** file for the tape drive containing the tape to any tape, regardless of the size specified in the tape's label. Make sure the tapes are at least as large as the tape size listed in the **TapeConfig** file. If a tape is larger, some of its capacity simply may not be used for the dump; if it is smaller, the dump may fail, but only after the Backup System fills the tape and determines that the tape is too small for the drive.

The Backup System does not require that a fileset fit entirely on a single tape. If the Tape Coordinator reaches the end of a tape while dumping a fileset, it puts the remaining data onto the next tape. The Backup Database automatically records that the fileset is on multiple tapes.

The **-noaction** option instructs the program to display a list of the filesets to be included in a dump set without actually performing the dump. This allows the issuer to determine how large the filesets are before actually dumping them; the issuer

can then better calculate the required number of tapes. The command ignores a value specified with the **-tcid** option if the **-noaction** option is used with the command.

The **bak restoreft**, **bak restoredisk**, and **bak restoreftfamily** commands can be used to restore data dumped with the **bak dump** command. You can use the commands to restore data to any type of file system (DCE LFS or non-LFS), regardless of the type of file system from which it was dumped. Thus, you can dump and restore data between DCE LFS and non-LFS file systems, and between different types of non-LFS file systems. (See the documentation for the **bak restoreft**, **bak restoredisk**, and **bak restoreftfamily** commands for more information about restoring data.

Note: On DCE 2.2 for AIX, backup and restore of non-LFS filesets is not supported.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines and in the **admin.ft** files on all File Server machines from which filesets are to be dumped.

OUTPUT

The following header is displayed in the command window followed by a list of the filesets, identified by name and fileset ID number, to be included in the dump set:

```
Preparing to dump the following filesets:list of filesets
```

The following message indicates that the Backup System has passed the dump request to the indicated Tape Coordinator:

```
Starting dump.
```

It is followed by a message that reports the unique dump ID number associated with this dump operation:

```
Dump ID of dump fileset_family_name.dump_level: dump_ID_number
```

The dump ID also appears in the Tape Coordinator monitoring window if the **butc** command is issued with debug level 1. The dump ID is not the same as the job ID number visible with **(bak) jobs** when **bak dump** is issued in interactive mode.

If the issuer includes the **-noaction** option, the output is

```
Starting dump of fileset family 'fileset family' (dump level 'dump level')
Total number of filesets : number
Would have dumped the following filesets:
      list of filesets
```

EXAMPLES

The following command dumps the filesets in the fileset family **user** according to the dump level **/full/week2/monday**. The issuer places the necessary tapes in the drive with TCID 5.

```
$ bak dump user /full/week2/monday 5
```

```
Preparing to dump the following filesets:
user.jones.bak 387623900
user.pat.bak 486219245
user.smith.bak 597315841
        .
        .
Starting dump.
Dump ID of dump user.monday: 34
```

The following command displays the list of filesets to be dumped when the **sys.rs_aix41** fileset family is dumped at the **/full** dump level:

```
$ bak dump sys.rs_aix41 /full -n
Starting dump of fileset family 'sys.rs_aix41' (dump level '/full')
Total number of filesets : 24
Would have dumped the following filesets:
      rs_aix41 124857238
      rs_aix41.bin 124857241
      rs_aix41.etc 124857246
         .         .
         .         .
```

RELATED INFORMATION

Commands: **bak adddump(8dfs)**, **bak addftentry(8dfs)**, **bak addftfamily(8dfs)**, **bak deletedump(8dfs)**, **bak dumpinfo(8dfs)**, **bak ftinfo(8dfs)**, **bak labeltape(8dfs)**, **bak lsdumps(8dfs)**, **bak readlabel(8dfs)**, **bak restoredisk(8dfs)**, **bak restoreft(8dfs)**, **bak rmdump(8dfs)**, **bak rmftfamily(8dfs)**. **bak restoreftfamily(8dfs)**,

bak dumpinfo

Purpose

Lists information about specified backups

Synopsis

bak dumpinfo{**-ndumps** *number* **-id** *dumpID*}[**-verbose**][**-help**]

OPTIONS

-ndumps *number*
Specifies the number of dumps about which information is to be displayed; information about the most recent number of dumps specified with this option is displayed. If fewer than the specified number of dumps exist, information about all existing dumps is displayed. Use this option or use **-id** ; omit both options to list information about the last 10 dumps.

-id *dumpID*
Specifies the unique dump ID number of a specific dump operation about which information is to be displayed. Use this option or use **-ndumps** ; omit both options to list information about the last 10 dumps.

-verbose
Includes a detailed list of information about the dump specified with the **-id** option. This option can be used only with **-id** .

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak dumpinfo** command lists information about specified dump sets. If a number is specified with **-ndumps**, information about that number of dump sets is displayed (information about the most recent **-ndumps** is displayed); if a specific dump ID number is specified with **-id**, information about that dump set is displayed; if both options are omitted, information about the 10 most recent dump sets is displayed.

The command displays information from the Backup Database. It can be used to display dump IDs prior to using the **bak deletedump** command to delete the record of one or more dump sets from the Backup Database. To view more detailed information about a specific dump set, specify both the **-id** option and the **-verbose** option.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

The following information is displayed for each dump listed:

DumpID
The dump set's ID number. This is a unique identifier that the Backup System uses internally.

parentID
The dump ID of the dump set that served as the parent for this dump set. A value of **0** (zero) means this is a full dump set and so has no parent, in which case **lvl** is also **0** (zero).

lvl The location in the dump hierarchy of the dump level used in creating the dump set. A value of **0** (zero) indicates a full dump set. A value of **1** or greater indicates an incremental dump set made at the indicated level in the hierarchy.

created
The date and time at which the Backup System started executing the dump operation that created this dump set.

nt The number of tapes required to record the dump set.

nfsets The number of filesets included in the dump set.

dump_name
The name of the dump set.

Additional information is displayed if both the **-id** and **-verbose** options are specified.

EXAMPLES

The following example displays information about the last three dumps:

```
$ bak dumpinfo -ndumps 3
    DumpID   parentID lvl     created     nt nfsets dump_name
-------------------------------------------------------------
 729293644  729289323   1  02/09/93  5:34  1     43 users.tue
 729287531  729286818   1  02/08/93  4:52  1     23 users.mon
 729286056          0   0  02/07/93  4:27  1     31 users.wk1
```

RELATED INFORMATION

Commands: **bak deletedump(8dfs)**, **bak dump(8dfs)**, **bak ftinfo(8dfs)**, **bak lsdumps(8dfs)**.

bak ftinfo

Purpose

Displays a fileset's dump history from the Backup Database

Synopsis

bak ftinfo -fileset *name* [**-help**]

OPTIONS

-fileset *name*
Names the fileset whose dump history is to be displayed. Include a **.backup** extension if the backup version of the fileset (rather than the read/write version) was dumped.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak ftinfo** command displays a dump history for the specified fileset, detailing the dates on which the fileset was cloned (the backup version was made) and dumped and the tapes on which it resides. If the dump was made of the backup version, as is usual, then **-fileset** must include the **.backup** extension.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

The output lists information about the dump sets in which **-fileset** is included, with the most recent dump set listed first. The output is displayed in six columns, as follows:

DumpID
The dump set's ID number. This is a unique identifier that the Backup System uses internally. It allows the issuer to check that the parent ID for an incremental dump set matches the dump ID of the dump set created previously.

parentID
The dump ID of the dump set that served as the parent for this dump set. A value of **0** (zero) means this is a full dump set and so has no parent, in which case **lvl** is also **0** (zero). It normally corresponds to the dump ID of the dump set created previously (the one on the next line of the output).

lvl The location in the dump hierarchy of the dump level used in creating the dump set. A value of **0** (zero) indicates a full dump set. A value of **1** or greater indicates an incremental dump set made at the indicated level in the hierarchy.

creation date
The date and time at which the Backup System started executing the dump operation that created the dump set.

clone date
The date and time at which the fileset was created. For a backup or read-only fileset, this represents the time at which it was cloned from its read/write source. For a read/write fileset, it indicates when the Backup System accessed the fileset to include it in the present dump set.

tape name
The name of the tape that contains the dump set.

EXAMPLES

The following command displays dump information about the fileset named *user.smith.backup*:

```
$ bak ftinfo user.smith.backup

   DumpID   parentID lvl creation date   clone date    tape name
654972910  654946323 1  10/01/91  5:07  10/01/91 4:01 users.tuesday.1
654960415  654946323 1  09/30/91  5:11  09/30/91 4:16 users.monday.1
654946323          0 0  09/29/91  5:36  09/28/91 4:31 users.week.1
```

RELATED INFORMATION

Commands: **bak dump(8dfs)**, **bak dumpinfo(8dfs)**, **bak lsdumps(8dfs)**.

bak help

Purpose

Shows syntax of specified **bak** commands or lists functional descriptions of all **bak** commands

Synopsis

bak help[**-topic** *string*][**-help**]

OPTIONS

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **dump**, not **bak dump**). If this option is omitted, the output provides a short description of all **bak** commands.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak help** command displays the first line (name and short description) of the online help entry for every **bak** command if -topic is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **bak apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

OUTPUT

The online help entry for each **bak** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with **Usage:**, lists the command options in the prescribed order.

EXAMPLES

The following command displays the online help entry for the **bak dump** command:

```
$ bak help dump
bak dump: start dump
Usage: bak dump -family <fileset_family_name> -level <dump_level>
[-tcid <tc_number>] [-noaction] [-help]
```

RELATED INFORMATION

Commands: **bak apropos(8dfs)**.

bak labeltape

Purpose

Creates the label on a tape

Synopsis

bak labeltape [**-tape** *tape_name*][**-size** *tape_size*][**-tcid** *tc_number*][**-help**]

OPTIONS

-tape *tape_name*
: Specifies the name to assign to the tape. If this option is omitted, the tape is marked as empty with a null identifier.

  An assigned name must reflect the dump set that is to go on the tape. It must be of the form *fileset_family_name.dump_level.index*, where *fileset_family_name* and *dump_level* constitute the name of the dump set to go on the tape. The *dump_level* is the last component of the name of the appropriate dump level; the *index* is an integer that represents the tape's place in the collection of tapes needed to contain the entire dump set. If the dump set fits on one tape, the index is 1.

-size *tape_size*
: Indicates the tape capacity. Providing this option is necessary only for information purposes. The Tape Coordinator uses the capacity specified in the **TapeConfig** file for any tape used in its tape drive. If this option is omitted, the size specified in the **TapeConfig** file for the drive is used for the tape's label.

  The default unit of size is kilobytes. It is also possible to express this number in megabyte or gigabyte units. To indicate megabyte units, add an uppercase or lowercase ″m″ with the number (with no space between the number and letter). To indicate gigabyte units, add an uppercase or lowercase ″g″ with the number (with no space between the number and letter).

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive containing the tape. If omitted, it defaults to **0** (zero).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak labeltape** command creates a label, readable by the Backup System, at the beginning of a tape. The issuer can either assign a name with the **-tape** option or omit the option to label the tape as empty.

The **-size** option is useful mainly for information purposes. The Tape Coordinator uses the capacity specified in the **TapeConfig** file for any tape used in its drive. It also copies the size specified in the **TapeConfig** file to the label of any tape that has no size specified in its label.

Labeling a tape is not a prerequisite to putting a dump set on it. The **bak dump** command accepts partially labeled or completely unlabeled tapes. However, the

bak labeltape command can be used to overwrite an existing label. This is useful if the data on a tape is no longer needed, but the tape's label prevents the tape from being used (because the label bears an inappropriate name or contains an unexpired expiration date). Overwriting the label with this command removes the obstacle to the tape's reuse.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command puts the label **user.monthly.1** on the tape in the drive whose TCID is **3**:

```
$ bak la user.monthly.1 -tcid 3
```

The following three commands are equivalent in effect. They all mark the tape in the drive whose TCID is **4** with a capacity of 2 gigabytes and the default name null.

```
$ bak label -size 2g -tcid 4
$ bak label -size 2048M -tcid 4
$ bak label -size 2097152 -tcid 4
```

RELATED INFORMATION

Commands: **bak readlabel(8dfs)**.

bak lsdumps

Purpose

Displays the dump hierarchy from the Backup Database

Synopsis

bak lsdumps[**-help**]

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **bak lsdumps** command displays the dump hierarchy from the Backup Database. A dump hierarchy can contain more than one full dump level, each of which defines a separate subhierarchy of dump levels. The **bak lsdumps** command displays the multiple subhierarchies if the Backup Database contains more than one full dump level.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

The output depicts the parent/child relationships between full and incremental dump levels in the dump hierarchy. The names of full dump levels are displayed at the far left margin. There can be more than one full dump in the hierarchy; each defines a subhierarchy of dump levels, each of which would presumably be used for dumping different fileset families.

Incremental dump levels are displayed below and indented to the right from their parent dump level, which can be either full or incremental. Incremental dump levels need not be directly below their parent; the amount of indentation alone indicates the parent/child relationship.

EXAMPLES

The following example displays a dump hierarchy with two subhierarchies. One subhierarchy starts with the full dump level **/month**, the other with the full dump level **/monday** (their positions at the left margin indicate they are full dump levels).

```
$ bak lsdumps
/month
      /week1
           /tuesday
                    /thursday
      /week2
           /tuesday
                    /thursday
/monday
       /tuesday
```

```
            /wednesday
                      /thursday
                                /friday
    /saturday
```

In the first subhierarchy, **/month** serves as the parent for the **/month/week1** and **/month/week2** dump levels, as indicated by the indentation (**/month/week2** is an example of how an incremental level need not be directly below its parent). The **/month/week1** dump level serves as the parent for the **/month/week1/tuesday** dump level, which serves as the parent for the **/month/week1/tuesday/thursday** level. These within-week relationships are repeated under **/month/week2**.

Dump sets created at the **/month** level are full dumps. Dumps performed at the **/month/week1** level include all files modified since the fileset family was dumped at the **/month** level. Dumps performed at the **/month/week1/tuesday** level include all files modified since the fileset family was dumped at the **/month/week1** level, and dumps done at the **/month/week1/tuesday/thursday** level include all files modified since the dump done at the **/month/week1/tuesday** level.

Dumps done at the **/month/week2** level would include all files modified since the fileset family was dumped at the **/month** level. Thus, dumps done at the**/month/week2** level serve as a summary of dumps done since the dump at the **/month/week1** level (they contain all files modified since a full dump was performed at the **/month** level).

The second subhierarchy, starting with **/monday**, is similarly constructed. The **/monday** dump level represents a full dump (it is at the far left margin); it is the parent for the **/monday/tuesday** level. The **/monday/tuesday** level is the parent for **/monday/tuesday/wednesday**, and so on. The **/monday/saturday** level's parent is **/monday**, so dumps performed at that level represent a summary of all the dumps performed at the intervening levels.

RELATED INFORMATION

Commands: **bak adddump(8dfs)**, **bak dump(8dfs)**, **bak dumpinfo(8dfs)**, **bak ftinfo(8dfs)**, **bak rmdump(8dfs)**.

bak lsftfamilies

Purpose

Lists fileset families defined in the Backup Database

Synopsis

bak lsftfamilies[**-family** *fileset_family_name*][**-help**]

OPTIONS

-family *fileset_family_name*
Names the fileset family to be displayed with the command. If omitted, all fileset families are displayed.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak lsftfamilies** command displays fileset family entry information about the specified fileset family. If -**family** is omitted, it lists all of the fileset families defined in the Backup Database. If **-family** is provided, it lists only that fileset family. In both cases, the fileset family entries in each fileset family are displayed.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

The output includes a separate entry for each fileset family. The entry lists the fileset family entries that make up the fileset family. Each fileset family entry is assigned an index number; the issuer of the **bak rmftentry** command uses these index numbers to identify the fileset family entries to delete.

EXAMPLES

The following command shows the fileset family entries in the two fileset families currently defined in the Backup Database:

```
$ bak lsftfamilies
Fileset family user:
    Entry 1: server.*, aggregate.*, filesets:user.*\.bak
Fileset family aix41:
    Entry 1: server.*, aggregate.*, filesets:aix41
```

RELATED INFORMATION

Commands: **bak addftentry(8dfs)**, **bak addftfamily(8dfs)**, **bak rmftentry(8dfs)**, **bak rmftfamily(8dfs)**.

bak lshosts

Purpose

Lists Tape Coordinator entries in the Backup Database

Synopsis

```
bak lshosts[-help ]
```

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **bak lshosts** command lists the Tape Coordinator entries currently defined in the Backup Database. The list includes the Tape Coordinators defined for all Tape Coordinator machines in the cell. Each Tape Coordinator is defined in the Backup Database and is, by implication, available for use. However, a Tape Coordinator process does not have to be running on a Tape Coordinator machine at the time this command is issued for its entry to be displayed.

Enter the **bak addhost** command to add an entry for a Tape Coordinator to the Backup Database. Enter the **bak rmhost** command to remove an entry for a Tape Coordinator from the Backup Database.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

The command first displays a **Tape hosts:** header. It then reports the following information for each Tape Coordinator:

- The name of the machine on which the Tape Coordinator is defined. (The format of the machine name depends on the form specified by the issuer of the **bak addhost** command.)
- The TCID of the Tape Coordinator. Valid TCIDs for Tape Coordinators are integers from 0 to 1023.

EXAMPLES

The following command displays the Tape Coordinators currently defined in the Backup Database:

```
$ bak lshosts
```

RELATED INFORMATION

Commands: **bak addhost(8dfs)**, **bak rmhost(8dfs)**.

bak readlabel

Purpose

Displays the name and size from a tape's label

Synopsis

bak readlabel[**-tcid** *tc_number*][**-help**]

OPTIONS

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive containing the tape. If omitted, it defaults to **0** (zero).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak readlabel** command displays the name and size from the label of the tape in the indicated tape drive. These values are placed on the tape with either the **bak dump** command or the **bak labeltape** command.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

For tapes with complete labels, a message appears listing the name and size of the tape. The tape name is of the form *fileset_family_name.dump_level.index*. If a tape has no name, the output reads **NULL**.

The tape size is expressed as follows: If an uppercase or lowercase ″g″ follows the size, it is a number of gigabytes; if an uppercase or lowercase ″m″ follows the size, it is a number of megabytes; if a lowercase ″k″ or the string **Kbytes** follows the size, it is a number of kilobytes.

If the tape is completely unlabeled or if the drive is empty, the output reads **Failed to read tape label**.

EXAMPLES

The following command shows the output for the tape with the label **sys.Monthly.3**. The capacity is 2 megabytes (expressed in kilobyte units). The tape is currently in the drive with a TCID of **6**.

```
$ bak read 6
Tape read was labelled : sys.Monthly.3 size : 2097152 Kbytes
```

The following command shows that the unlabeled tape in the drive with a TCID of **0** (zero) has a capacity of 5 gigabytes:

```
$ bak read
Tape read was labelled : NULL size : 5G
```

RELATED INFORMATION

Commands: **bak dump(8dfs)**, **bak labeltape(8dfs)**.

bak restoredb

Purpose

Restores a backup copy of the Backup Database

Synopsis

bak restoredb[**-tcid** *tc_number*][**-help**]

The alias is **bak dbrestore**.

OPTIONS

-tcid *tc_number*
: Specifies the TCID of the Tape Coordinator for the tape drive from which the Backup Database is to be restored. If omitted, it defaults to **0** (zero).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak restoredb** command restores a backup copy of the entire Backup Database. If the Backup Database becomes damaged, you should delete the old database; then use this command to restore an entirely new version from its backup tape (which must be named **bak_db_dump.1**). The Backup Database is damaged if the disk housing the database becomes damaged or the **bak verifydb** command fails.

Do not attempt to recover information from a corrupted database. Instead, stop all **bakserver** processes and remove the old Backup Database from each machine on which it is located.

After the database is removed, restart all **bakserver** processes on the machines on which they were running. Use the **bak addhost** command to add a tape host for the Tape Coordinator from which you plan to restore the Backup Database. Then use the **bak restoredb** command to restore the new version of the database; the **-tcid** option specifies the TCID of the Tape Coordinator from which to restore the Backup Database (the Tape Coordinator just added with the **bak addhost** command).

Use the **bak savedb** command to save the Backup Database to tape.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

RELATED INFORMATION

Commands: **bak savedb(8dfs)**, **bak verifydb(8dfs)**.

bak restoredisk

Purpose

Restores the contents of an entire aggregate from tape

Synopsis

bak restoredisk -server *machine* **-aggregate** *name*[**-tcid** *tc_number*][**-newserver** *machine*][**-newaggregate** *name*][**-noaction**][**-help**]

The alias is **bak dkrestore**.

OPTIONS

-server *machine*
Names the File Server machine that houses the aggregate you want to restore. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name or aggregate name of the aggregate on the machine indicated with the **-server** option that you want to restore. These names are specified in the first and second fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive in which you are placing the necessary tapes. If omitted, it defaults to **0** (zero).

-newserver *machine*
Names the File Server machine to which to restore the data. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option only if the destination server is different from the server specified with the **-server** option.

-newaggregate *name*
Specifies the device name or aggregate name of the aggregate to which to restore the data. These names are specified in the first and second fields of the entry for the aggregate in the **dfstab** file. Use this option only if the name of the destination aggregate is different from the name of the aggregate specified with the **-aggregate** option.

-noaction
Directs the command to display the list of tapes necessary to perform the indicated restore without actually performing the operation.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak restoredisk** command restores the contents of the aggregate specified with the **-server** and **-aggregate** options to the file system. To do this, the **bak** program contacts the Fileset Location Server (FL Server) for a listing from the Fileset Location Database (FLDB) of all the filesets that reside on the specified

aggregate. It then consults the Backup Database to learn which tapes contain the full and incremental dumps needed to restore every fileset from the aggregate. This command is useful if a disk or machine failure destroys the data on an entire aggregate.

To restore filesets from the specified aggregate to the same site (the site specified with the **-server** and **-aggregate** options), omit the **-newserver** and **-newaggregate** options. The data in the restored filesets overwrites the filesets' current contents; there is no change in the Fileset Location Database (FLDB) entries for the filesets.

To restore the filesets to an alternate site, include the **-newserver** option, the **-newaggregate** option, or both. The filesets continue to use their existing FLDB entries and fileset ID numbers, and the filesets' FLDB entries are updated to record the new site. The current contents of each fileset are replaced with the data restored from tape. The command allows you to restore filesets to a new site as follows:

- To restore the filesets to a different aggregate on the same File Server machine, specify the new aggregate with the **-newaggregate** option.
- To restore the filesets to an aggregate of the same name on a different File Server machine, specify the new File Server machine with the **-newserver** option.
- To restore the filesets to a completely different site, specify the new File Server machine with the **-newserver** option and the new aggregate with the **-newaggregate** option.

If you specify a new site and the filesets to be restored currently exist at their old site, you must use the **fts zap** command to delete the existing filesets before issuing the **bak restoredisk** command. The **bak restoredisk** command fails if you do not use the **fts zap** command to delete the existing filesets before using the **bak restoredisk** command to restore the filesets to the new site.

Note: Do not use the **fts delete** command to delete the existing filesets and their FLDB entries before issuing the **bak restoredisk** command. If you use the **fts delete** command instead of the **fts zap** command, you cannot use the **bak restoredisk** command to restore the filesets; you can restore the filesets only with the **bak restoreft** command.

Note: To restore after a disk crashes

1. Replace your disk and create the new aggregate using the same aggregate ID as the one that was lost.
2. Export the new aggregate.
3. Use the command **bak restoredisk**.

The **-noaction** option instructs the command to produce a list of the tapes the Backup System would need to perform the indicated restore without actually performing the operation. To do so, include the **-noaction** option with all of the other options to be used with the actual command.

Data can be dumped and restored between different types of file systems. For example, data dumped from a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of nonLFS fileset; likewise, data dumped from a nonLFS fileset can be restored to a DCE LFS fileset or to a different type of nonLFS fileset.

Note: On DCE 2.2 for AIX, backup and restore of non-LFS filesets is not supported.

Restored data is translated into the appropriate format for the file system to which it is restored. Note that incompatible information may be lost when a fileset is dumped and restored between different types of file systems. For example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Use the **bak restoreft** command to restore one or more filesets to a single site. Use the **bak restoreftfamily** command to restore a fileset family or to restore one or more filesets to the same site or to different sites.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines and in the **admin.ft** file on the File Server machine to which filesets are to be restored.

OUTPUT

If you do not include the **-noaction** option, the **bak restoredisk** command returns the unique dump ID number associated with the restore operation. The dump ID is displayed in the command window following the command line and in the Tape Coordinator's monitoring window if the **butc** command is issued with debug level 1. The dump ID is not the same as the job ID number visible with the **(bak) jobs** command if the **bak restoredisk** command is issued in interactive mode.

If you include the **-noaction** option, a list of filesets to be restored is displayed followed by a list of the tapes necessary to complete the restore operation. No dump ID number is reported because none is assigned.

EXAMPLES

The following command restores the filesets listed in the FLDB as residing on the aggregate named **/dev/lv01** on the File Server machine named **fs5**. The filesets are restored to the same aggregate and server machine. Tapes are placed in the drive with a TCID of **3**.

```
$ bak restored /.../abc.com/hosts/fs5 /dev/lv01 3

Starting restore
bak: dump ID of restore operation: 253
bak: Finished doing restore
```

The following command restores the filesets listed in the FLDB as stored on the aggregate named **/dev/lv02** on the File Server machine named **fs1**. The filesets are restored to a new site, the aggregate **/dev/lv01** on the File Server machine **fs3**. The **fts zap** command is used to delete existing filesets from the current site before the **bak restoredisk** command is issued. Tapes are placed in the drive with a TCID of **0** (zero).

```
$ bak restored /.../abc.com/hosts/fs1 /dev/lv02 -news/.../abc.com/hosts/fs3 \
-newa /dev/lv01

Starting restore
bak: dump ID of restore operation: 256
bak: Finished doing restore
```

RELATED INFORMATION

Commands: **bak dump(8dfs)**, **bak restoreft(8dfs)**, **bak restoreftfamily(8dfs)**, **fts delete(8dfs)**, **fts zap(8dfs)**.

Files: **dfstab(4dfs)**.

bak restoreft

Purpose

Restores filesets from tape

Synopsis

bak restoreft -server *machine***-aggregate** *name* **-fileset** *name* [**-extension** *name_extension*][**-date** *date*][**-tcid** *tc_number*][**-noaction**][**-help**]

The alias is **bak ftrestore**.

OPTIONS

-server *machine*
Names the File Server machine to which to restore each specified fileset. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. If the fileset currently exists at a site other than the one specified with this option and the **-aggregate** option, you must delete the existing fileset before restoring it to the specified site.

-aggregate *name*
Specifies the device name or aggregate name of the aggregate to which to restore each specified fileset. These names are specified in the first and second fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file. If the fileset currently exists at a site other than the one specified with this option and the **-server** option, you must delete the existing fileset before restoring it to the specified site.

-fileset *name*
Names each fileset to be restored. Provide the name of the read/write version of each fileset, even if (because of its fileset entry definition in a fileset family) the backup version of a fileset was actually dumped. The command automatically appends a **.backup** extension to the name of a fileset if it can find no record in the Backup Database of a backup performed for the fileset's read/write version.

-extension *name_extension*
Specifies an extension to add to the restored fileset's name to distinguish it from a fileset of the same name that currently exists in the file system. This causes the Backup System to restore the data from tape into a new fileset independent of the existing one. Any string other than **.readonly** or **.backup** is acceptable; if a period is to precede the extension, include it in the string provided.

-date *date*
Specifies the date prior to which a dump must have been made to be included in the restore. The **-date** option indicates a date-specific restore; only dump sets dated before the specified date are restored. If omitted, this option defaults to **0** (zero) and a full restore of the most recently dumped version of the fileset occurs. Otherwise, there are two types of legal values:

*mm***/***dd***/***yy*
Specifies 00:00 (12:00 a.m.) on the indicated date. A value of this type causes a date-specific restore containing only data from dumps done before the indicated date (for example, **11/22/91**).

mm/*dd*/*yy hh*:*mm*

Specifies a time on the indicated date. A value of this type causes a date-specific restore containing only data from dumps done before the indicated date and time. The time must be in 24-hour format (for example, **20:30** is 8:30 p.m.). Surround the entire argument with " " (double quotes) because it contains a space.

-tcid *tc_number*

Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive in which you are placing the necessary tapes.

-noaction

Directs the command to produce the list of tapes necessary to perform the indicated restore without actually performing the operation.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak restoreft** command restores the contents of each fileset indicated with the **-fileset** option from tape to the indicated site (File Server machine and aggregate). By default, restores are full, recreating the fileset as it existed when it was last dumped. A full restore includes data from the last full dump and all subsequent incremental dumps (if any). If incremental dumps exist, you are prompted to insert the necessary tapes into the tape drive. To have the command produce a list of the tapes that the Backup System would need to perform the indicated restore without actually performing the operation, include the **-noaction** option with the command.

You can also choose to do a date-specific restore by including the **-date** option. A date-specific restore returns the fileset to the state it was in at its last dump before the indicated date. Rather than including all dumps to the final one done, it includes only the last full dump and any incremental dumps done before the indicated date.

The precise effect of a restore depends on whether the fileset currently exists in the file system and whether you want to preserve its current state. To replace the current contents of a fileset with data restored from tape, omit the **-extension** option. The results are as follows:

- If the **-server** and **-aggregate** options specify the fileset's current site, the restored data overwrites the fileset's current contents. There is no change in the Fileset Location Database (FLDB) entry for the fileset.
- If the **-server** and **-aggregate** options specify a new site, the restored data is stored in a new fileset at the indicated site. If you name a new site and the fileset to be restored currently exists at its old site, you must do one of the following before issuing the command:
  - Use the **fts zap** command to delete the existing fileset. The fileset continues to use its existing FLDB entry and fileset ID number, and the fileset's FLDB entry is updated to record the new site.
  - Use the **fts delete** command to delete the existing fileset and its FLDB entry. The fileset receives a new FLDB entry and a new fileset ID number.

  Using the **fts zap** command is the better approach because it preserves the fileset's existing ID number, which allows Cache Managers to continue to access the fileset without updating their tables of mappings between fileset names and fileset ID numbers. The **bak restoreft** command fails if you do not use the **fts**

zap or **fts delete** command to delete the existing fileset before using the **bak restoreft** command to restore the fileset to the new site.

To preserve a fileset's current contents but also introduce a restored version into the file system, use the **-extension** option. A new fileset at the site specified with the **-server** and **-aggregate** options then contains the restored data. It has the same name as the current fileset, with the addition of the distinguishing extension. The Fileset Location (FL) Server automatically assigns the new fileset a fileset ID number and a new FLDB entry, which records all of the appropriate information about the new fileset.

You can also restore a fileset that no longer exists in the file system. A new fileset at the site specified with the **-server** and **-aggregate** options is created to contain the restored data.

Data can be dumped and restored between different types of file systems. For example, data dumped from a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of nonLFS fileset; likewise, data dumped from a nonLFS fileset can be restored to a DCE LFS fileset or to a different type of nonLFS fileset.

Note: On DCE 2.2 for AIX, backup and restore of non-LFS filesets is not supported.

Restored data is translated into the appropriate format for the file system to which it is restored. Note that incompatible information may be lost when a fileset is dumped and restored between different types of file systems. For example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Use the **bak restoredisk** command to restore the contents of an entire aggregate. Use the **bak restoreftfamily** command to restore a fileset family or to restore one or more filesets to the same site or to different sites.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines and in the **admin.ft** file on the File Server machine to which filesets are to be restored.

CAUTIONS

Overwriting an existing fileset destroys any files created in the current fileset after the date of the last dump included in the restore. It is always safer to preserve the current fileset by using the **-extension** option to restore data to a new fileset.

OUTPUT

If you do not include the **-noaction** option, the **bak restoreft** command returns the unique dump ID number associated with the restore operation. The dump ID is displayed in the command window directly following the command line and in the Tape Coordinator's monitoring window if the **butc** command is issued with debug level 1. The dump ID number is not the same as the job ID number visible with the **(bak) jobs** command if the **bak restoreft** command is issued in interactive mode.

If you include the **-noaction** option, a list of filesets to be restored is displayed, followed by a list of the tapes necessary to complete the restore operation. No dump ID number is reported because none is assigned.

EXAMPLES

The following command restores the fileset named *user.pat* to the aggregate named **/dev/lv01** on the File Server machine named */.../abc.com/hosts/fs5*:

```
$ bak restoreft /.../abc.com/hosts/fs5 /dev/lv01 user.pat
Starting restore
bak: dump ID of restore operation: 187
bak: Finished doing restore
```

RELATED INFORMATION

Commands: **bak dump(8dfs)**, **bak restoredisk(8dfs)**, **bak restoreftfamily(8dfs)**, **fts delete(8dfs)**, **fts zap(8dfs)**.

Files: **dfstab(4dfs)**.

bak restoreftfamily

Purpose

Restores a fileset family or one or more specified filesets from tape

Synopsis

bak restoreftfamily{**-family** *fileset_family_name* **-file** *filename*}[**-tcid** *tc_number*] [**-noaction**][**-help**]

The alias is **bak familyrestore**.

OPTIONS

-family *fileset_family_name*
Specifies a fileset family to be restored. The command restores all of the filesets in each of the fileset entries in the specified fileset family. Refer to the section entitled **Using the -family Option** for information about using this option. Either this option or the **-file** option must be specified.

-file *filename*
Specifies the full pathname of a file from which the command is to read the name of each fileset to be restored and the site (File Server machine and aggregate) to which the fileset is to be restored. Specify each fileset and site on a separate line, using the following format:

machine aggregate fileset

Refer to the section entitled **Using the -file Option** for information about using this option. Either this option or the **-family** option must be specified.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive in which you are placing the necessary tapes. If this option is omitted the TCID defaults to 0 (zero).

-noaction
Directs the command to produce a list of filesets it would restore without actually restoring the filesets. The command also provides additional information, such as the tapes that contain dumps of the filesets and the sites to which the filesets would be restored. Include the other options as you would to actually execute the command. You can use this option with the **-family** option to write a list of filesets to a file, which you can then modify for use with the **-file** option. See the section of this reference page entitled **Output** for information about using the **-noaction** option.

-help Prints help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak restoreftfamily** command restores the contents of specified filesets from tape to the file system. The command performs a full restore of each indicated fileset, restoring data from the last full dump and all subsequent incremental dumps (if any) of each fileset. Use the **-family** option or the **-file** option to indicate the filesets to be restored, as follows:

- The **-family** option lets you restore all of the filesets included in the fileset entries in a specified fileset family. The command reads the Fileset Location Database (FLDB) to determine the filesets to be restored and restores them to their current sites.
- The **-file** option lets you restore specific individual filesets that have entries in a specified file. The command restores each fileset to the site you specify.

The **-noaction** option instructs the command to produce a list of the filesets it would restore without actually restoring any filesets. The command also provides information about the tapes that contain dumps of the filesets. You can use the **-noaction** option with the **-file** option to determine the tapes required to restore the indicated filesets. You can also use the **-noaction** option with the **-family** option to construct a list of filesets that would be restored with a specified fileset family; you can then modify the list of filesets as necessary to produce a file for use with the **-file** option.

The **bak restoreftfamily** command is useful for recovering from catastrophic losses of data, such as the loss of all filesets on multiple aggregates of a File Server machine or the loss of multiple aggregates from multiple File Server machines. In such cases, the command provides a better approach to recovery than the **bak restoreft** command or the **bak restoredisk** command because

- It allows you to restore either individual filesets or specialized collections of filesets.
- It allows you to restore different filesets to different sites.

Conversely, the **bak restoreft** command restores one or more filesets to a single site, and the **bak restoredisk** command restores all filesets that reside on a single aggregate to a single aggregate. The **bak restoreftfamily** command provides greater breadth to a restore operation than the other commands that restore data, which instead provide convenient depth.

Regardless of the command used, data can be dumped and restored between different types of file systems. For example, data dumped from a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of non-LFS fileset; likewise, data dumped from a non-LFS fileset can be restored to a DCE LFS fileset or to a different type of non-LFS fileset.

Note: On DCE 2.2 for AIX, backup and restore of non-LFS filesets is not supported.

Restored data is translated into the appropriate format for the file system to which it is restored. Note that incompatible information may be lost when a fileset is dumped and restored between different types of file systems. For example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Using the -family Option

Use the **-family** option of the **bak restoreftfamily** command to restore the filesets included in a fileset family. The command reads the FLDB to determine all filesets that satisfy the fields of the entries in the specified fileset family. It then looks in the Backup Database to determine the tapes that contain the last full dump and all subsequent incremental dumps of each fileset. It restores each fileset included in an entry in the fileset family to its current site, overwriting an existing version of the fileset.

You can specify the name of an existing fileset family, or you can define a new fileset family and add entries that correspond to the filesets that need to be restored. For example, suppose you need to restore all filesets that reside on the File Server machines named **fs1.abc.com** and **fs2.abc.com**. You can use the **bak addftfamily** command to create a new fileset family. You can then use the **bak addftentry** command to add the following entries to the new fileset family:

```
/.../abc.com/hosts/fs1.* .*
/.../abc.com/hosts/fs2.*.*
```

These entries indicate all filesets on all aggregates on the machines named **fs1.abc.com** and **fs2.abc.com**. Once the new fileset family is defined, you can issue the **bak restoreftfamily** command, specifying the name of the fileset family with the command's **-family** option.

When you create fileset families for use with the **bak restoreftfamily** command, define entries that match the read/write versions of filesets. The command automatically appends a **.backup** extension to the name of a fileset if it can find no record in the Backup Database of a backup performed for the read/write version. You can include a **.backup** extension to match the backup versions of filesets only if the backup versions were dumped to tape and the backup versions are still valid in the FLDB entries for the filesets.

Using the -file Option

Use the **-file** option of the **bak restoreftfamily** command to restore each fileset that has an entry in a specified file. The command examines the Backup Database to determine the tapes that contain the last full dump and all subsequent incremental dumps of each specified fileset and each fileset to the site indicated in the specified file. It does not consult the FLDB.

An entry for a fileset in a file to be used with the command must have the following format:

machine aggregate fileset [*comments...*]

The entry provides the following information:

machine
: Specifies the File Server machine to which the fileset is to be restored. Identify the machine by its DCE pathname (for example, */.../abc.com/hosts/fs1*), its host name (for example, **fs1.abc.com**), or its IP address (for example, **11.22.33.44**).

aggregate
: Specifies the aggregate to which the fileset is to be restored. Identify the aggregate by its device name (for example, **/dev/lv01**) or by its aggregate name (for example, **lfs1**). These names are specified in the first and second fields of the entry for the aggregate in the *dcelocal***/var/dfs/dfstab** file.

fileset
: Specifies the fileset to be restored. Specify the name of the read/write version of the fileset, even if the backup version of the fileset was actually dumped. The command automatically appends a **.backup** extension to the name of the fileset if it can find no record in the Backup Database of a backup performed for the read/write version. (Note that you can specify the name of the backup version of the fileset if the backup version was dumped to tape.)

comments...

All remaining text. The command treats any other text provided with the entry for the fileset as a comment and ignores it. Any additional text is optional.

Do not use wildcards (for example, **.***) in an entry. Also, do not include a newline character in an entry. Each entry must appear on a single line of the file. The command uses only the first line for a given fileset; it ignores all subsequent lines for the fileset.

If you restore a fileset to the site at which it currently exists, the command overwrites the existing version of the fileset. If you restore a fileset to a site other than the site at which it currently exists, you must do one of the following before issuing the command:

- Use the **fts zap** command to delete the existing fileset. The restored fileset continues to use its existing FLDB entry and fileset ID number, and the fileset's FLDB entry is updated to record the new site.
- Use the **fts delete** command to delete the existing fileset and its FLDB entry. The restored fileset receives a new FLDB entry and a new fileset ID number.

Using the **fts zap** command is the better approach because it preserves a fileset's existing ID number, which allows Cache Managers to continue to access the fileset without updating their tables of mappings between fileset names and fileset ID numbers. The **bak restoreftfamily** command fails if you do not use the **fts zap** or **fts delete** command to delete an existing fileset before using the **bak restoreftfamily** command to restore the fileset to a new site.

Privileges Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines and in the **admin.ft** file on each File Server machine to which one or more filesets are to be restored.

OUTPUT

If you do not include the **-noaction** option, the **bak restoreftfamily** command returns the unique dump ID number associated with the restore operation. The dump ID is displayed in the command window directly following the command line and in the Tape Coordinator's monitoring window if the **butc** command is issued with debug level 1. The dump ID number is not the same as the job ID number visible with the **(bak) jobs** command if the **bak restoreftfamily** command is issued in interactive mode. Note that the dump ID and job ID numbers are not assigned to the operation until the command actually begins to restore filesets.

If you include the **-noaction** option, the command displays on standard output the number of filesets that would be restored, followed by a separate line of information about each fileset. For each fileset, the command provides the following output:

```
machine aggregate fileset_dumped # as fileset_restored; tape tape_name; pos
position_number; date
```

The output provides the following information:

machine

The host name of the File Server machine to which the fileset would be restored (for example, **fs1.abc.com**).

aggregate
The aggregate name of the aggregate to which the fileset would be restored (for example, **lfs1**).

fileset_dumped
The name of the fileset that was dumped (for example, *user.frost*). The command can display the name of the backup version of the fileset (for example, *user.frost.backup*) if that version was dumped.

fileset_restored
The name with which the fileset would be restored (for example, *user.frost*). The command always displays the name of the read/write version of the fileset.

tape_name
The name of the tape that contains the dump set with which the fileset was dumped (for example, **user.full.1**).

position_number
The position of the fileset with respect to other filesets on the tape that contains the dump set (for example, **31**).

date The date and time at which the fileset was dumped (for example, **Wed Jul 13 05:59:01 1994**).

The command displays information only for filesets that have been dumped to tape; for each fileset that has not been dumped, the command displays an error message on standard error output. The command reads the Backup Database to determine everything but the *machine*, *aggregate*, and *fileset_dumped* information. If you use the **-noaction** option with the **-file** option, the *machine*, *aggregate*, and *fileset_dumped* information must be provided in the specified file; if you use the **-noaction** option with the **-family** option, the command examines the FLDB to determine this information, so it provides information only for filesets that have entries in the FLDB.

The command displays multiple lines of information for a fileset if one or more incremental dumps were performed since the last full dump of the fileset. The command displays one line of output for the last full dump and one line of output for each incremental dump. It displays the lines in the order in which the dumps would need to be restored, beginning with the full dump. It does not necessarily present all of the lines for a fileset consecutively.

If you intend to write the output of the **-family** and **-noaction** options to a file for use with the **-file** option, include only a single line for each fileset; the command ignores any additional lines for a fileset. You can include any line for the fileset; all lines name the fileset's current site. You do not need to remove the # (number sign) and the information that follows it; the command ignores any characters that follow the third argument on a line.

When the **-noaction** option is included, no dump ID and job ID numbers are reported because none are assigned.

The amount of time required for the **bak restoreftfamily** command to complete depends on the number of filesets to be restored. However, a restore operation that includes a large number of filesets can take hours to complete. To reduce the amount of time required for the operation, you can execute multiple instances of the command simultaneously, specifying disjoint fileset families with each command if you use the **-family** option, or indicating files that list different filesets with each

command if you use the **-file** option. Depending on how the filesets to be restored were dumped to tape, specifying disjoint fileset families can also enable you to make the most efficient use of your backup tapes when many filesets need to be restored.

EXAMPLES

The following command restores all filesets included in entries in the fileset family **data.restore**, which was created expressly to restore data to a pair of File Server machines on which all data was corrupted due to a software error. All filesets are restored to the sites recorded in their entries in the FLDB.

```
$ bak restoreftfam data.restore

Starting restore
bak: dump ID of restore operation: 112
bak: Finished doing restore
```

The following command restores all filesets that have entries in the file named **/tmp/restore**:

```
$ bak restoreftfam -file /tmp/restore

Starting restore
bak: dump ID of restore operation: 113
bak: Finished doing restore
```

The file **/tmp/restore** has the following contents:

```
/.../abc.com/hosts/fs1 /dev/lv01 user.abhijit
/.../abc.com/hosts/fs1 /dev/lv01 user.vijay
/.../abc.com/hosts/fs1 /dev/lv01 user.pierette
/.../abc.com/hosts/fs2 /dev/lv01 user.frost
/.../abc.com/hosts/fs2 /dev/lv01 user.wvh
          .            .       .
          .            .       .
```

RELATED INFORMATION

Commands: **bak addftentry(8dfs)**, **bak addftfamily(8dfs)**, **bak dump(8dfs)**, **bak restoredisk(8dfs)**, **bak restoreft(8dfs)**, **fts delete(8dfs)**, **fts zap(8dfs)**

Files: **dfstab(4dfs)**

bak rmdump

Purpose

Deletes a dump level from the Backup Database

Synopsis

bak rmdump -level *dump_level*[**-help**]

OPTIONS

-level *dump_level*
Names the dump level to be deleted; specify the complete pathname for the dump level to be removed, including any necessary / (slashes).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak rmdump** command deletes the dump level indicated with the **-level** option from the dump hierarchy in the Backup Database. If the dump level is a parent, all dump levels that are its children (and their children, if any) are also deleted.

EXAMPLES

The following command deletes the dump level called **week3** from the dump hierarchy:

```
$ bak rmd /week3
```

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

RELATED INFORMATION

Commands: **bak adddump(8dfs)**, **bak dump(8dfs)**, **bak lsdumps(8dfs)**.

bak rmftentry

Purpose

Deletes a fileset family entry from a fileset family

Synopsis

bak rmftentry -family *fileset_family_name* **-entry** *fileset_entry_index* [**-help**]

OPTIONS

-family *fileset_family_name*
Names the fileset family from which to delete the entry.

-entry *fileset_entry_index*
Identifies the fileset family entry to delete. The legal value is the fileset entry index number, a positive integer. The **bak lsftfamilies** command displays the index number of each fileset family entry in a fileset family (the first entry defined has index 1, the second 2, and so on).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak rmftentry** command deletes the indicated fileset family entry from the fileset family specified with **-family**. Use **-entry** to identify the fileset family entry by its index number.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command deletes the fourth fileset family entry from the fileset family called **sys**. The issuer first used the **bak lsftfamilies** command to determine that the index number of the fileset family entry to be deleted is 4.

```
$ bak rmfte sys 4
```

RELATED INFORMATION

Commands: **bak addftentry(8dfs)**, **bak lsftfamilies(8dfs)**.

bak rmftfamily

Purpose

Deletes a fileset family from the Backup Database

Synopsis

bak rmftfamily -family *fileset_family_name*...[**-help**]

OPTIONS

-family *fileset_family_name*
Names each fileset family to be deleted.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak rmftfamily** command deletes each fileset family specified by **-family** from the Backup Database. It also deletes the fileset family entries contained in each deleted family. The **bak addftfamily** command is used to add fileset families to the Backup Database.

Use the **bak lsftfamilies** command to list the fileset families currently defined in the Backup Database. Use the **bak rmftentry** command to remove a currently defined fileset family entry from the Backup Database.

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command deletes the fileset family called **user**:

```
$ bak rmftf user
```

RELATED INFORMATION

Commands: **bak addftfamily(8dfs)**, **bak lsftfamilies(8dfs)**, **bak rmftentry(8dfs)**.

bak rmhost

Purpose

Removes a Tape Coordinator entry from the Backup Database

Synopsis

bak rmhost[**-tcid** *tc_number*][**-help**]

OPTIONS

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator to be removed. Legal values are integers from 0 to 1023. If this option is omitted, a value of **0** (zero) is used.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak rmhost** command deletes the indicated Tape Coordinator entry from the Backup Database. The Backup Server no longer sends requests to that Tape Coordinator, even if it is still operational on the machine. Repeat this command once for each Tape Coordinator whose entry is to be deleted.

The mapping between the TCID of a Tape Coordinator and the device name of the drive with which it is associated is recorded in the **TapeConfig** file on the Tape Coordinator machine. Remove the entry for a Tape Coordinator from the **TapeConfig** file when you remove its entry from the Backup Database.

Enter the **bak addhost** command to add an entry for a Tape Coordinator to the Backup Database. Enter the **bak lshosts** command to list the Tape Coordinators that have entries in the Backup Database.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command removes the entry for the Tape Coordinator with a TCID of **1** from the Backup Database:

```
$ bak rmhost 1
```

RELATED INFORMATION

Commands: **bak addhost(8dfs)**, **bak lshosts(8dfs)**.

Files: **TapeConfig(4dfs)**.

bak savedb

Purpose

Creates a backup copy of the Backup Database

Synopsis

bak savedb[**-tcid** *tc_number*][**-help**]

OPTIONS

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive to which the database is to be backed up. If omitted, it defaults to **0** (zero).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak savedb** command creates a backup copy of the entire Backup Database. Designate one tape as the Backup Database tape; label it with the name **bak_db_dump.1** (it must have this name). The **-tcid** option specifies the TCID of the Tape Coordinator to which to save the Backup Database; this option is necessary only if the TCID is not **0** (zero).

If the Backup Database is damaged, delete the old database and use the **bak restoredb** command to restore a new version from tape. Use the **bak verifydb** command to determine if the Backup Database is damaged.

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command backs up the Backup Database to a tape in the Tape Coordinator with a TCID of **3**:

```
$ bak save 3
```

RELATED INFORMATION

Commands: **bak restoredb(8dfs)**, **bak verifydb(8dfs)**.

bak scantape

Purpose

Extracts dump set information from a tape

Synopsis

bak scantape[**-dbadd**][**-tcid** *tc_number*][**-help**]

OPTIONS

-dbadd
Adds the information extracted from the tape to the Backup Database if the tape is completely undamaged and the Backup Database does not already contain an entry with the same dump ID number.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive containing the tape. If omitted, it defaults to **0** (zero).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak scantape** command reads the tape in the drive controlled by the Tape Coordinator indicated by **-tcid**, extracting information from the tape label and from the fileset header of each fileset on the tape. It does not extract actual data from the filesets, though the information it does extract is needed to restore the data using the Backup System.

The Tape Coordinator displays the information about each fileset in its monitoring window as it extracts the information. The Tape Coordinator checks for damage to the tape medium by checking for the presence of special markers it expects to find at the start and end of each fileset. If the Tape Coordinator does not find an expected marker, it concludes that the tape medium is damaged, and the command aborts.

If the **-dbadd** option is provided, the program creates a Backup Database entry for the tape and records the extracted information in the entry. It is not possible to extract information about only specific filesets on a tape. Because only data about all of the filesets on a tape can be extracted, this command works only if a tape is completely undamaged.

The Tape Coordinator does not require that the issuer insert all of the tapes containing a dump set unless a fileset is split across two tapes. In that case, it automatically prompts for the tape with the next highest index to extract complete information about the fileset. If **-dbadd** is used, information from both tapes is added to the database.

If a tape contains only complete filesets, the Tape Coordinator reads the tape and prompts

```
Are there more tapes? (y/n)
```

If the issuer responds **n**, the command exits, adding the information from the tape to the Backup Database if **-dbadd** is used. If the issuer responds **y**, the Tape Coordinator prompts for the tape with the next highest index.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

CAUTIONS

Using the **-dbadd** option with this command introduces the possibility that two database entries will appear almost the same; you will need to track which physical tape corresponds to which entry.

Database entries are identified by three elements: the tape name, the dump level pathname, and a dump ID number, which is unique for every dump set. This command creates a database entry for the dump set on the tape as long as its dump ID number is different from any existing entry's ID number, even if the entry has the same tape name and dump level name as an existing entry.

OUTPUT

The **bak scantape** command first displays the following information from the label of the tape:

name The tape label, in the format *fileset_family_name.dump_level.index*.

createTime
The date and time at which the Backup System started executing the dump operation that created this dump set.

expire time
The date when this tape expires.

cell The cell in which the dump set was created.

size The tape's capacity in kilobytes (not the amount of data on the tape). The value comes from the tape label and is derived from **bak labeltape** or the **TapeConfig** file rather than from a measurement of the tape.

dump path
The dump level used in creating the dump set.

dumpID
The dump ID number of the dump on the tape.

useCount
The number of times data has been dumped to this tape.

The command then displays an entry for each fileset. The entries appear in the order in which the filesets are encountered on the tape. If a fileset is split across two tapes, there is a separate entry for both fragments. Each entry includes the following information:

fileset name
The name of the fileset, with a **.backup** or **.readonly** extension if appropriate.

fileset ID
The fileset ID number of the fileset.

dumpSetName
The dump set to which the fileset belongs.

dumpID
The dump ID number of the dump set named by **dumpSetName**.

level The depth in the dump hierarchy of the dump level used in creating the dump set. A value of **0** (zero) indicates a full dump set. A value of **1** or greater indicates an incremental dump set made at the indicated depth in the hierarchy. The value reported is for the entire dump, not necessarily for the fileset itself. (For example, it is possible for an individual fileset to be dumped at a higher level if it was omitted from a previous dump set.)

parentID
The dump ID number of **dumpSetName**'s parent dump set. (A parent dump set is a dump set made at the level that serves as the parent for a dump level.) This should be **0** (zero) if **level** is **0** (zero).

endTime
Should always be **0** (zero); it is for internal use only.

clonedate
The date and time at which the fileset was created. For a backup or read-only fileset, this represents the time when it was cloned from its read/write source fileset. For a read/write fileset, it indicates when the Backup System accessed the fileset to include it in **dumpSetName**.

The following error message (usually preceded by other, more specific messages) indicates that the program has not encountered one of the markers it expects to find at the start and end of each fileset and has concluded that the tape is damaged. No data from this tape can be incorporated into the Backup Database.

```
aborting - this dump cannot be processed correctly
```

EXAMPLES

The following command shows the output from a tape's label and for the first fileset on the tape:

```
$ bak scantape
Tape label
----------
name =         guests.monthly.1
createTime =    Fri Nov 22 05:59:31 1990
expireTime =    Mon Nov 23 05:59:31
1990

cell =         /.../abc.com
size =          20103324 Kbytes
dump path =     /monthly
dump id =       729369701
useCount =      1
-- End of tape label --
-- fileset --
fileset name: user.guest10.backup
fileset ID 0,,112262
dumpSetName: guests.monthly
dumpID 729369701
level 0
parentID 0
endTime 0
clonedate Fri Nov 22 05:36:29 1991
```

RELATED INFORMATION

Commands: **bak deletedump(8dfs)**, **bak dump(8dfs)**, **bak restoredisk(8dfs)**, **bak restoreft(8dfs)**, **kill** (see the **bak(8dfs)** reference page for information about the **kill** command).

bak setexp

Purpose

Sets the expiration date on an existing dump level

Synopsis

bak setexp -level *dump_level*[**-expires** *date*][**-help**]

OPTIONS

-level *dump_level*

Names each dump level whose expiration date is to be set. Provide the full pathname for each dump level, including all necessary / (slashes).

-expires *date*

Defines the expiration date to be associated with each dump level. Expiration dates can be specified as absolute or relative values. Absolute expiration dates have the format

at *mm*/*dd*/*yy* [*hh*:*mm*]

The word **at** is followed by a date (*month*/ *day*/ *year*) and, optionally, a time (*hours*: *minutes*). Values that can be interpreted for *yy* run from 00 to 37, which are interpreted as the years 2000−2037, and from 70 to 99, which are interpreted as 1970−1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038−2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038.

If provided, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). If omitted, the time defaults to **00:00** (12:00 a.m.).

Relative expiration dates have the format

in [*integer***y**] [*integer***m**] [*integer***d**]

The word **in** is followed by a number of years (maximum 9999), months (maximum 11), and days (maximum 30), or a combination of these arguments. At least one of the three must be provided, and the appropriate unit abbreviation (**y**, **m**, or **d**) must always accompany a value. If more than one of the three is provided, they must appear in the order shown. As with absolute dates, a number of years that causes the relative time to exceed the year 2038 is effectively truncated to the number of years remaining until 2038.

If you omit this option, tapes created at the specified dump levels have no expiration dates, meaning they can be overwritten by appropriately named dump sets at any time. Although the **-expires** option is followed by an ellipsis, you can specify only one expiration date. (The ellipsis is included to accommodate the DFS command parser.)

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak setexp** command sets the expiration date on each dump level specified with **-level**. Each dump level must already exist in the dump hierarchy stored in the Backup Database.

The expiration date is applied to tapes containing dump sets made at the dump level; after the specified date, the Backup System overwrites a tape's contents with acceptably named dump sets without question. The Backup System's attempts to overwrite an unexpired tape fail until the issuer relabels the tape with the **bak labeltape** command. (Because the label records the unexpired expiration date or unacceptable name, erasing the label removes the obstacle to overwriting.) If no expiration date is defined for a tape, the Backup System overwrites the dump set on the tape with a dump set of the same name without question.

Expiration dates can be either absolute or relative:

1. Absolute expiration dates are defined as a specific month/day/year and, optionally, hours and minutes. A tape with an absolute expiration date expires at that time, regardless of when the dump set on it was created. (If the expiration predates the dump set's creation, the tape is immediately treated as expired.)
2. Relative dates are defined as a number of years, months, days, or any combination of the three. When the Backup System creates a dump set at the dump level, it calculates the tape's actual expiration date by adding the relative date to the start time of the dump operation.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

EXAMPLES

The following command associates an absolute expiration date of 10:00 p.m. on 31 December 1990 with the dump level **/90/december**:

```
$ bak setexp /90/december -e at 12/31/90 22:00
```

The following command associates a relative expiration date of 7 days with the two dump levels **/monthly/week1** and **/monthly/week2**:

```
$ bak set /monthly/week1 /monthly/week2 -exp 7d
```

RELATED INFORMATION

Command: **bak adddump(8dfs)**, **bak dump(8dfs)**, **bak labeltape(8dfs)**.

bak status

Purpose

Reports on the operation that a Tape Coordinator is executing

Synopsis

bak status[**-tcid** *tc_number*][**-help**]

OPTIONS

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for which status information is to be displayed.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak status** command displays information about the operation currently being performed by the indicated Tape Coordinator. The command displays information about the Tape Coordinator's current job as well as any pending jobs waiting for the Tape Coordinator.

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

If the indicated Tape Coordinator is not currently performing an operation, the output reports **Tape coordinator is idle**. Otherwise, it reports the following:

1. An operation name describing the operation. One of the following operation names is displayed:

   Dump (*dump_set***)**
   : For a dump operation, where *dump_set* is the name of the dump set in the form *fileset_family_name.dump_level*. Dump operations are initiated with the **bak dump** command.

   Restore
   : For a restore operation. Restore operations are initiated with the **bak restoreft**, **bak restoredisk**, or **bak restoreftfamily** command.

   Labeltape (*tape_label***)**
   : For a tape labeling operation, where *tape_label* is the label being placed on the tape. Tape labeling operations are started with the **bak labeltape** command.

   Scantape
   : For a tape scanning operation. Tape scanning operations are initiated with the **bak scantape** command.

   SaveDb
   : For a database saving operation. Operations that save the Backup Database to tape are started with the **bak savedb** command.

RestoreDb
For a database restoring operation. Operations that restore the Backup Database from tape are initiated with the **bak restoredb** command.

2. The number of kilobytes transferred so far (from file system to tape for a dump operation, from tape to file system for a restore operation).
3. The string **fileset** followed by the name of the fileset currently being restored if the operation is a restore; the string **fileset** followed by the name of the fileset currently being dumped if the operation is a dump.
4. Status information about the operation. If the operation is executing normally, no message is displayed; otherwise, one of the following messages is displayed:

[abort request rcvd]
The **kill** command was issued, the Tape Coordinator received the **kill**, but the operation is not yet canceled.

[abort complete]
The Tape Coordinator has completed the kill request.

[callout in progress]
The Tape Coordinator is waiting for a callout routine to complete.

[drive wait]
The Tape Coordinator is in the process of locking the device.

[done]
The Tape Coordinator completed the operation.

[operator wait]
The Tape Coordinator is waiting for the operator monitoring the operation to insert a tape in the drive.

EXAMPLES

The following command displays status information about the operation being performed by the Tape Coordinator with a TCID of **4**. The operation is a dump of the dump set whose name is **usersys.monday**. So far, 23,597 bytes have been dumped to tape. The fileset named *user.terry* is currently being dumped. No status message is displayed, indicating the operation is proceeding normally.

```
$ bak status 4
Dump (usersys.monday): 23597 Kbytes transferred, fileset user.terry.
```

RELATED INFORMATION

Commands: **bak(8dfs)**, **bak dump(8dfs)**, **bak labeltape(8dfs)**, **bak restoredb(8dfs)**, **bak restoredisk(8dfs)**, **bak restoreft(8dfs)**, **bak restoreftfamily(8dfs)**, **bak savedb(8dfs)**, **bak scantape(8dfs)**,

bak verifydb

Purpose

Checks the status of the Backup Database

Synopsis

bak verifydb[**-verbose**][**-help**]

OPTIONS

-verbose
Directs the command to provide more information about the Backup Database.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bak verifydb** command checks the status of the Backup Database. It displays a message indicating whether the Backup Database is undamaged or damaged. If the Backup Database is undamaged, it can be accessed. If it is damaged, it must be restored from tape with the **bak restoredb** command (provided it has been backed up previously with the **bak savedb** command).

Privilege Required

The issuer must be listed in the **admin.bak** files on all Backup Database machines.

OUTPUT

Depending on the condition of the Backup Database, this command displays one of the following two messages:

Database OK.
Indicates that the database is undamaged and can be used.

Database not OK.
Indicates that the database is damaged. The database must be deleted and then restored from tape.

If the **-verbose** option is included with the command, the command reports some additional information about the Backup Database. One reason to use the **-verbose** option is to determine if your Backup Database has any orphan blocks, which are blocks that it preallocated but cannot use. Orphan blocks are not a problem for the database. However, if you are concerned with disk usage on the machine on which the database resides, you can eliminate the unusable blocks by saving the database to tape with the **bak savedb** command and then restoring it with the **bak restoredb** command.

The **-verbose** option also causes the command to display the name of the machine on which the command is issued.

EXAMPLES

The following command verifies that the Backup Database is undamaged:

```
$ bak verifydb
Database OK.
```

RELATED INFORMATION

Commands: **bak dumpinfo(8dfs)**, **bak ftinfo(8dfs)**, **bak lsdumps(8dfs)**, **bak restoredb(8dfs)**, **bak savedb(8dfs)**.

bakserver

Purpose

Initializes the Backup Server

Synopsis

bakserver[**adminlist** *filename*][**-verbose**][**-help**]

OPTIONS

-adminlist *filename*
: Specifies the file that contains principals and groups authorized to execute **bakserver** RPCs (usually using **bak** commands). If this option is omitted, the **bakserver** obtains the list of authorized users from the default administrative list file, *dcelocal***/var/dfs/admin.bak**.

-verbose
: Directs the **bakserver** process to provide a detailed report on what it is doing by displaying messages on standard error.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **bakserver** command. See the **bos help** and **bos apropos** pages for examples of using these commands.

DESCRIPTION

The Backup Server (**bakserver** process) communicates with the Backup Database to perform dump and restore operations. The **bakserver** process must run on all machines that house a copy of the Backup Database. It is usually started and controlled by the BOS Server; if it is not, execute the **bakserver** process as a background process. The binary file for the **bakserver** process resides in *dcelocal***/bin/bakserver**.

The first time it is initialized, the **bakserver** process creates the Backup Database in *dcelocal***/var/dfs/backup**; all Backup Database files have a root name of **bkdb**. The **bakserver** process also creates the *dcelocal***/var/dfs/admin.bak** administrative list file if the file does not already exist.

The principals and members of groups in the **admin.bak** administrative list are authorized to issue **bak** commands (which are used for tasks such as examining the database and dumping and restoring data). The list must also include the abbreviated DFS server principals of all Backup Database machines to allow for the proper distribution of changes via the Ubik database synchronization mechanism.

Because the Backup Database is a replicated database, the **admin.bak** administrative lists for all **bakserver** processes in a cell must contain the same principals and groups.

It is recommended that all system administrators using the Backup System be included on the following lists: the **admin.bak** file on all machines housing the Backup Database, the **admin.fl** file on all machines housing the Fileset Location Database (FLDB), and the **admin.ft** file on all File Server machines.

When it is started, the **bakserver** process makes a **ubik_ServerInit** call to register its existence as a server process with the Ubik coordinator. It then listens for incoming RPCs to which to respond.

Each time it is started, the **bakserver** process also creates the *dcelocal***/var/dfs/adm/BakLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when the **bakserver** process is started, the process moves it to the **BakLog.old** file in the same directory (overwriting the current **BakLog.old** file if it exists) before creating a new version to which to append messages.

Privilege Required

The issuer must be logged in as **root** on the local machine.

OUTPUT

If problems are encountered during initialization, the **bakserver** process displays error messages on standard error output. The **bakserver** process keeps an event log in the file *dcelocal***/var/dfs/adm/BakLog**.

If run with the **-verbose** option, the **bakserver** process provides a detailed report on what it is doing by displaying messages on standard error.

RELATED INFORMATION

Files: **admin.bak(4dfs)**, **BakLog(4dfs)**.

bos

Purpose

Introduction to the **bos** command suite

OPTIONS

The following options are used with many **bos** commands. They are also listed with the commands that use them.

-server *machine*

Names the machine running the BOS Server that is to execute the command. To run a privileged **bos** command (a **bos** command that requires the issuer to have some level of administrative privilege) using a privileged identity, always specify the full DCE pathname of the machine (for example, */.../abc.com/hosts/fs1*).

To run an unprivileged **bos** command, you can use any of the following to specify the machine:

- The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*)
- The machine's host name (for example, **fs1.abc.com** or **fs1**)
- The machine's IP address (for example, **11.22.33.44**)

Note: If you specify the host name or IP address of the machine, the command executes using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option); unless DFS authorization checking is disabled on the specified machine, a privileged **bos** command issued in this manner fails. If you specify the machine's host name or IP address, the command displays the following message (using the **-noauth** option suppresses the message):

```
bos:  WARNING: short form for server used;
no authentication information will be sent to
the bosserver
```

-noauth

Directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. Generally, the -noauth option is included with a command if DFS authorization checking is disabled on the server machine whose BOS Server is to execute the command or if the Security Service is unavailable. If DFS authorization checking is disabled, the BOS Server requires no administrative privilege to issue any command; any user, even the identity **nobody**, has sufficient privilege to perform any operation. If the Security Service is unavailable, a user's security credentials cannot be obtained.

DFS authorization checking is disabled with the **bos setauth** command or by including the **-noauth** option when the **bosserver** process is started on a machine. DFS authorization checking is typically disabled

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file

Include the **-noauth** option with a command that requires administrative privilege only if DFS authorization checking is disabled on the necessary machine. A command that requires administrative privilege fails if the **-noauth** option is included and DFS authorization checking is not disabled. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, */.../abc.com/hosts/fs1/dfs-server*. (Do not confuse a machine's DFS server principal with its unique **self** identity.)

Use this option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the -noauth option.

-help Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see the **dfs_intro(8dfs)** reference page.

DESCRIPTION

Commands in the **bos** command suite are used by system administrators to contact the Basic OverSeer (BOS) Server. The BOS Server runs on every DFS server machine to monitor the other DFS server processes on the machine. It restarts processes automatically if they fail. The BOS Server also provides an interface through which system administrators can start and stop processes and check on server status.

The files described in the following sections are used to store configuration, administrative, and security information.

The BosConfig File

The *dcelocal***/var/dfs/BosConfig** file on the local disk of each DFS server machine contains information about the processes the BOS Server is to monitor. This information includes the process type, the command parameters associated with the process, and a status flag that tells the BOS Server to start the process at initialization or restart the process if the process fails. Whenever the BOS Server starts or restarts, it reads the file to learn which processes to monitor; this information is transferred into memory and the file is not read again until the BOS Server next restarts.

The administrator can change the process status in the BOS Server's memory with specific **bos** commands; therefore, it is possible for a process to stop running even if its status flag in the BosConfig file is set to **Run**. Similarly, an administrator can start a process without setting its status flag in the **BosConfig** file to **Run** by changing its memory state flag to **Run**.

Never edit the **BosConfig** file directly; always use the appropriate **bos** commands. Editing the file directly can introduce changes of which the BOS Server is unaware. The BOS Server does not recognize such changes until it is restarted and again reads the file.

The admin.bos File

The *dcelocal*/**var/dfs/admin.bos** file on the local disk of each File Server machine contains the names of users who are allowed to issue **bos** commands on that machine. All users can list the contents of the file with the **bos lsadmin** command; only administrative users can edit the contents of the file with the **bos addadmin** and **bos rmadmin** commands. Because the **admin.bos** file is a binary file, you cannot edit it directly; you must use the appropriate **bos** commands.

The Keytab File

A **/krb5/v5srvtab** keytab file is stored on the local disk of each File Server machine. A keytab file contains the list of server encryption keys used by a server process on that machine to decrypt tokens presented by clients. The server process interacts only with clients possessing tokens encrypted with server encryption keys listed in the appropriate keytab file.

The keys in a keytab file are marked with a unique key version number. All tokens presented by clients are also marked with a key version number; a server process uses the key version number to determine which key to use to decrypt a token.

Only administrative users can examine, add, and remove keys in the keytab file. Never edit a keytab file directly; always use the appropriate **bos** commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ man bos
: Displays the reference page for the command suite.

$ man bos_ *command*
: Displays the reference page for an individual command. You must use an _ (underscore) to connect the command suite to the command name. *Do not use the underscore when issuing the command in DFS.*

$ bos help
: Displays a list of commands in a command suite.

$ bos help *command*
: Displays the syntax for a single command.

$ bos apropos -topic *string*
: Displays a short description of any commands that match the specified *string*.

Consult the **dfs_intro(8dfs)** reference page for complete information about the DFS help facilities.

Privilege Required

All **bos** commands can be issued by users listed in the **admin.bos** file on the machine whose BOS Server is executing the command. Specific privilege information is listed with each command's description. In addition, if the BOS Server is running with DFS authorization checking disabled, no privilege is required to issue **bos** commands.

CAUTIONS

Never directly edit a **BosConfig** file, a keytab file, an **admin.bos** file, or any administrative (**admin**) file; always use the appropriate commands from the **bos** command suite.

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos addkey(8dfs)**, **bos apropos(8dfs)**, **bos create(8dfs)**, **bos delete(8dfs)**, **bos gckeys(8ds)**, **bos genkey(8dfs)**, **bos getdates(8dfs)**, **bos getlog(8dfs)**, **bos getrestart(8dfs)**, **bos help(8dfs)**, **bos install(8dfs)**, **bos lsadmin(8dfs)**, **bos lscell(8dfs)**, **bos lskeys(8dfs)**, **bos prune(8dfs)**, **bos restart(8dfs)**, **bos rmadmin(8dfs)**, **bos rmkey(8dfs)**, **bos setauth(8dfs)**, **bos setrestart(8dfs)**, **bos shutdown(8dfs)**, **bos start(8dfs)**, **bos startup(8dfs)**, **bos status(8dfs)**, **bos stop(8dfs)**, **bos uninstall(8dfs)**, **dfs_intro(8dfs)**, **keytab(8dce)**.

Files: **admin.bak(4dfs)**, **admin.bos(4dfs)**, **admin.fl(4dfs)**, **admin.ft(4dfs)**, **admin.up(4dfs)**, **BosConfig(4dfs)**, **v5srvtab(5sec)**.

bos addadmin

Purpose

Adds a user, group, or server to an administrative list

Synopsis

```
bos addadmin -server  machine -adminlist  filename [-principal name...]
 [-group name][-createlist][-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*
: Names the server machine that houses the administrative list to which principals, groups, or both are to be added. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-adminlist *filename*
: Names the administrative list to which principals, groups, or both are to be added. The complete pathname is unnecessary if the list is stored in the default configuration directory (*dcelocal***/var/dfs**).

-principal *name*
: Specifies the principal name of each user or server machine to be added to the administrative list. A user from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** *cellname***/** *username* or just *username*); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** *cellname***/hosts/** *hostname***/self** or just **hosts/** *hostname***/self**); a server machine from a foreign cell can be specified only by a full principal name.

-group *name*
: Specifies the name of each group to be added to the administrative list. A group from the local cell can be specified by a full or abbreviated group name (for example, ***/.../*** *cellname***/** *group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

-createlist
: Specifies that the file indicated with **-adminlist** is to be created if it is not found. Any principals or groups specified with the command are added to the new file; if no principals or groups are specified, the command creates an empty file. This option has no effect if the specified file already exists.

  Note: Because the **admin.bos** list must already exist to issue this command, this option is ignored if **admin.bos** is specified with the **-adminlist** option.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos addadmin** command adds the specified users, groups, and servers to the administrative list specified by the **-adminlist** option on the server machine indicated by the **-server** option. The principal (login) names of users and the principal names of server machines to be added to the administrative list are specified with the **-principal** option; the names of groups to be added to the list are specified with the **-group** option. Principals added to the administrative list either directly (with the **-principal** option) or indirectly (as members of groups indicated with the **-group** option) can then issue administrative commands for the DFS server process associated with the list.

The default path for administrative lists is the configuration directory (*dcelocal*/**var/dfs**). If the specified list is stored in the default directory, only the specific filename is required. If the specified list is stored elsewhere, the pathname to the file that was used when the associated server process was started is required.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command adds the user names jones and smith to the **admin.bos** file on **fs1**. The administrative list is stored in the default configuration directory.

```
$ bos adda -server /.../abc.com/hosts/fs1 -adminlist admin.bos -principal
jones smith
```

RELATED INFORMATION

Commands: **bos lsadmin(8dfs)**, **bos rmadmin(8dfs)**.

Files: **admin.bak(4dfs)**, **admin.bos(4dfs)**, **admin.fl(4dfs)**, **admin.ft(4dfs)**, **admin.up(4dfs)**.

bos addkey

Purpose

Converts a string into a server encryption key and adds it to a keytab file

Synopsis

```
bos addkey-server machine -kvno +_or_version_number -password string
[-principal name] [-localonly ] [-noauth | -localauth][-help ]
```

OPTIONS

-server *machine*

Names the server machine whose keytab file is to have a new key added to it. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-kvno *+_or_version_number*

Defines the key version number of the new key. The version number must be one of the following:

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the principal indicated by **-principal** in the keytab file on the server machine specified by **-server**.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the principal indicated by **-principal** in the Registry Database. However, it may not be unique for the indicated principal in the keytab file on the specified machine, in which case it replaces the key currently associated with the principal/version number pair in the keytab file.

Unless the **-localonly** option is used, the new key and its version number replace the key and version number currently stored in the Registry Database for the indicated principal.

-password *string*

Defines a character string to be converted into an octal string for use as the key. The string serves as a password for the indicated principal. It can include any characters; it can also include spaces if the entire string is enclosed in " " (double quotes).

-principal *name*

Provides the principal name with which the key is to be associated. The default is the DFS principal name of the machine specified by **-server**.

-localonly

Specifies that the key is to be added to the keytab file on the machine indicated by **-server**, but that the Registry Database is not to be updated. The default is to add the key to the local keytab file and update the Registry Database accordingly.

-noauth

Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS

authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos addkey** command associates a new server encryption key with the principal name indicated by **-principal** in the **/krb5/v5srvtab** keytab file on the server machine specified by **-server** and, by default, in the Registry Database. The key is derived from the string specified by -password and is given the version number specified by **-kvno**. The issuer can either specify a version number or have the command choose one that is unique for the indicated principal in the Registry Database. If the **-localonly** option is omitted, the server encryption key and version number for the indicated principal are automatically updated both in the keytab file on the specified server machine and in the Registry Database; if the **-localonly** option is specified, the keytab file is updated, but the Registry Database is not.

The **bos genkey** command is a more secure way of adding a key to a keytab file because it generates a random key. It also always updates the Registry Database. The keytab file must already exist before the **bos addkey** or **bos genkey** command can be used to add a key to it. (Keytab files are created with the **dcecp keytab create** command.)

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**, and, unless the **-localonly** option is used, the DFS server principal of the machine specified by **-server** must have the permissions necessary to alter entries in the Registry Database.

OUTPUT

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS Server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

EXAMPLES

The following command adds a new server encryption key with key version number **14** to the keytab file on **fs1** without updating the Registry Database. Because **-principal** is omitted, the key is associated with the DFS principal name of **fs1** (the machine specified with -server). The password string **fourteenth new key** is converted into an octal key before being placed in the keytab file.

```
$ bos addk /.../abc.com/hosts/fs1 14 "fourteenth new key" -localonly
```

RELATED INFORMATION

Commands: **bos gckeys(8dfs)**, **bos genkey(8dfs)**, **bos lskeys(8dfs)**, **bos rmkey(8dfs)**, **keytab(8dce)**.

Files: **v5srvtab(5sec)**.

bos apropos

Purpose

Shows each help entry containing a specified string

Synopsis

bos apropos -topic *string* [**-help**]

OPTIONS

-topic *string*
Specifies the keyword string for which to search. If it is more than a single word, surround the string with " " (double quotes) or other delimiters. Type all strings for **bos** commands in lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos apropos** command displays the first line of the help entry for any **bos** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **bos help** command.

Privilege Required

No privileges are required.

OUTPUT

The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **bos** command where the string specified by **-topic** is part of the command name or the first line.

EXAMPLES

The following command lists all **bos** commands that have the word **restart** in their names or short descriptions:

```
$ bos apropos restart
getrestart: get restart times
restart: restart all processes
setrestart: set restart times for server processes
```

RELATED INFORMATION

Commands: **bos help(8dfs)**.

bos create

Purpose

Creates a new process in the **BosConfig** file and starts it

Synopsis

```
bos create -server machine -process server_process -type process_type -cmd cmd_line
 [-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*
: Names the server machine on which to create the new process. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-process *server_process*
: Names the server process to be created. You can choose any name for a process, but it is recommended that you give the process the same name as its binary file (and use the same name on every machine running that process). The recommended names are

  ftserver
  : For the Fileset Server process

  flserver
  : For the Fileset Location Server process

  upclient
  : For the client portion of the Update Server, which brings common configuration files and binary files from the System Control and Binary Distribution machines

  upserver
  : For the server portion of the Update Server process

  repserver
  : For the Replication Server process

  bakserver
  : For the Backup Server process

  Each process runs under the local identity **root** and the DCE identity **self**. However, the process is unauthenticated as far as DFS is concerned.

-type *process_type*
: Specifies the process type. Legal values are **simple** and **cron**. Specify **simple** for continuous processes and **cron** for processes that are to run only at specified times.

-cmd *cmd_line*
: Specifies the commands the BOS Server runs to start the process and, if -type is **cron**, the time the BOS Server executes the command.

  For a **simple** *process*, this must be the complete pathname to the binary file for the process (for example, */.../dceloca**l**/**bin/flserver** for the Fileset

Location Server). The commands for some **simple** processes take options, in which case the entire argument must be surrounded with " " (double quotes).

For a **cron** *process*, provide two parameters. The first parameter is either the pathname to a binary file to be executed or the complete pathname of a command from one of the DFS suites (complete with all of the necessary arguments). Surround this parameter with " " (double quotes) if it contains spaces.

If the specified executable file does not exist, the **bos create** command does not create an entry in the **BosConfig** file. Instead, the command displays the following message:

```
bos:  failed to create a new server instance instance of
type process_type (specified executable not found)
```

The second parameter for a **cron** process specifies the time when the BOS Server is to execute the command specified by the first parameter. Use a day and time together to execute the command weekly at the specified time; use a time alone to execute the command daily at the specified time. Day and time specifications have the following format:

[*day***]** *hh***:***mm*

Enter the name of the day in all lowercase letters, giving either the whole name or the first three letters as an abbreviation (for example, **sunday** or **sun**). Specify the time of day by separating the hours from the minutes with a **:** (colon). Use 24-hour time (for example, **14:30**), or use 1:00 to 12:00 with **am** or **pm** (for example, "**2:30 pm**"). The time part of the option is optional if the day is specified; if the time is excluded, it defaults to 00:00 on the specified day. As shown in the example, enclose the entire entry in " " (double quotes) if it contains a space.

To execute the command only once, specify **now** instead of a day or a day and time, or issue the command directly; the process entry is removed from the **BosConfig** file after the command is executed. To place the process entry in the **BosConfig** file without ever executing it, specify **never** instead of a time or a day and time.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos create** command creates a new server process on the server machine specified by **-server** by creating an entry in the **BosConfig** file on the local disk of

the machine. The status of the new process entry in both the **BosConfig** file and memory is set to **Run**, and the process is started on the server machine (unless the process is a **cron** process and the second parameter of the **-cmd** option is **never**).

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command creates the **simple** process **flserver** on the machine named **fs3**:

```
$ bos create /.../abc.com/hosts/fs3 flserver simple/dcelocal/bin/flserver
```

The following command creates the **cron** process named **backup** on the machine named **fs3**. The **-localauth** option allows the process (which runs unauthenticated) to use the DFS server principal of **fs3** to execute the privileged **fts clonesys** command.

```
$ bos create /.../abc.com/hosts/fs3 backup cron "/dcelocal/bin/fts clonesys \
-s /.../abc.com/hosts/fs3 -localauth" 5:30
```

RELATED INFORMATION

Commands: **bos delete(8dfs)**.

Files: **BosConfig(4dfs)**.

bos delete

Purpose

Deletes server processes from the **BosConfig** file

Synopsis

bos delete -server*machine* **-process** *server_process* [**-noauth** | **-localauth**] [**-help**]

OPTIONS

-server *machine*
Names the server machine from which to delete one or more server processes. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-process *server_process*
Names each process to delete. Use the name assigned with the **-process** option in the **bos create** command; if necessary, use the **bos status** command to list the possible process names.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos delete** command removes each indicated server process entry from the **BosConfig** file on the server machine specified by **-server**. Before issuing this command, the issuer must use the **bos stop** command to stop each indicated process, both **simple** and **cron**, running on **-server**. An error message results if the status flag of a process being deleted is **Run** when this command is issued.

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command removes the **flserver** process entry from the **BosConfig** file on the machine named **fs3**:

```
$ bos delete /.../abc.com/hosts/fs3 flserver
```

RELATED INFORMATION

Commands: **bos create(8dfs)**.

Files: **BosConfig(4dfs)**.

bos gckeys

Purpose

Removes obsolete server encryption keys from a keytab file

Synopsis

bos gckeys -server *machine*[**-principal** *name*][**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine whose keytab file is to have obsolete keys removed from it. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-principal *name*
Provides the principal name for which obsolete keys are to be removed from the keytab file. The default is the DFS principal name of the machine specified by **-server**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos gckeys** command removes obsolete server encryption keys from the **/krb5/v5srvtab** keytab file on the server machine specified by **-server**. Obsolete keys associated only with the principal name specified by **-principal** are removed from the keytab file; the DFS principal name of the server machine specified with **-server** is used by default.

Keys are removed based on age and lack of use. The removal process, referred to as *garbage collection*, affects only the keytab file stored on the local disk of the machine indicated by **-server** ; it has no effect on the Registry Database.

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**.

OUTPUT

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS Server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

EXAMPLES

The following command removes obsolete keys associated with the principal **hosts/fs1/dfs-server** from the keytab file on the server machine named */.../abc.com/hosts/fs3*. Note that the keys being removed are associated with the principal name of a machine different from the one whose BOS Server is executing the command.

```
$ bos gckeys/.../abc.com/hosts/fs3 hosts/fs1/dfs-server
```

RELATED INFORMATION

Commands: **bos addkey(8dfs)**, **bos genkey(8dfs)**, **bos lskeys(8dfs)**, **bos rmkey(8dfs)**, **keytab(8dce)**.

Files: **v5srvtab(5sec)**.

bos genkey

Purpose

Generates a random key and adds it to a keytab file

Synopsis

```
bos genkey -server machine -kvno +_or_version_number [-principal name]
[-noauth | -localauth] [-help ]
```

OPTIONS

-server *machine*
Names the server machine whose keytab file is to have a new key added to it. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-kvno *+_or_version_number*
Defines the key version number of the new key. The version number must be one of the following:

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the principal indicated by **-principal** in the keytab file on the server machine specified by **-server**.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the principal indicated by **-principal** in the Registry Database. However, it may not be unique for the indicated principal in the keytab file on the specified machine, in which case it replaces the key currently associated with the principal/version number pair in the keytab file.

The new key and its version number always replace the key and version number currently stored in the Registry Database for the indicated principal.

-principal *name*
Provides the principal name with which the key is to be associated. The default is the DFS principal name of the machine specified by **-server**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos genkey** command associates a new server encryption key with the principal name indicated by **-principal** in the **/krb5/v5srvtab** keytab file on the server machine specified by **-server** and in the Registry Database. The command generates a random key and assigns it the version number indicated by **-kvno**. The issuer can either specify a version number or have the command choose one that is unique for the indicated principal in the Registry Database. The server encryption key and version number for the specified principal are automatically updated both in the keytab file on the specified server machine and in the Registry Database.

The **bos addkey** command can also be used to add a key to a keytab file with or without updating the Registry Database. However, it is less secure because the issuer must specify a string to be converted into the server encryption key. The keytab file must already exist before the **bos genkey** or **bos addkey** command can be used to add a key to it. (Keytab files are created with the **dcecp keytab create** command.)

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**, and the DFS server principal of the machine specified by **-server** must have the permissions necessary to alter entries in the Registry Database.

OUTPUT

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS Server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

EXAMPLES

The following command generates a new server encryption key with key version number **14** and adds it to the keytab file on **fs1**. Because -principal is omitted, the key is associated with the DFS principal name of **fs1** (the machine specified with **-server**). The Registry Database is updated automatically.

```
$ bos genkey /.../abc.com/hosts/fs1 14
```

RELATED INFORMATION

Commands: **bos addkey(8dfs)**, **bos gckeys(8dfs)**, **bos lskeys(8dfs)**, **bos rmkey(8dfs)**, **keytab(8dce)**.

Files: **v5srvtab(5sec)**.

bos getdates

Purpose

Lists time stamps on versions of binary files

Synopsis

bos getdates -server *machine* **-file** *binary_file* [**-dir** *alternate_dest*] [**-noauth** | **-localauth**] [**-help**]

OPTIONS

-server *machine*
Names the server machine that houses the binary files whose time stamps are to be displayed. The BOS Server on this machine executes the command. Specify the machine's DCE pathname, its host name, or its IP address.

-file *binary_file*
Names the current version of each binary file whose time stamps are to be displayed. The time stamps on the current, **.BAK**, and **.OLD** versions of each file are displayed. All specified files must reside in the same directory (*dcelocal***/bin**, by default, or an alternate directory specified with the **-dir** option). Specify only filenames; if a pathname is provided for a file, the command ignores all but the final element.

-dir *alternate_dest*
Provides the pathname of the directory in which all specified files reside. Omit this option if the files reside in the default directory, *dcelocal***/bin**; otherwise, provide a full or relative pathname. Relative pathnames (pathnames that do not begin with a slash) are interpreted relative to the *dcelocal* directory on the machine specified by **-server**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos getdates** command displays the time stamps for the current, **.BAK**, and **.OLD** versions of each binary file whose current version is specified with the **-file** option. The time stamps record when the files were installed. The command displays a message for any version of a specified file that does not exist. Use the

-server option to specify the name of the server machine on which the files reside. The **-dir** option can be used to specify the name of the directory in which the files reside if it is different from *dcelocal***/bin**.

The BOS Server automatically creates **.BAK** and **.OLD** versions when new binaries are installed with **bos install**. Use the **bos uninstall** command to replace the current version with its next-oldest version (**.BAK** or, if the **.BAK** version does not exist, **.OLD**) or to remove all versions of a binary file. Use the **bos prune** command to remove **.BAK** and **.OLD** files from the *dcelocal***/bin** directory. (This command can also be used to remove core files from the *dcelocal***/var/dfs/adm** directory.)

Privilege Required

No privileges are required.

OUTPUT

For each file specified with the **-file** option, the output reports the time stamps on the current, **.BAK**, and **.OLD** versions. The output displays a message to indicate any version that does not exist.

EXAMPLES

The following command displays the time stamps on the three versions of the **flserver** binary file stored in the default directory on the server machine named **fs2**:

```
$ bos getdates /.../abc.com/hosts/fs2 flserver
```

RELATED INFORMATION

Command: **bos install(8dfs)**, **bos prune(8dfs)**, **bos uninstall(8dfs)**.

bos getlog

Purpose

Examines the log file for a server process

Synopsis

bos getlog *machine*-file *log_file*[**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine from which to retrieve the log file. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-file *log_file*
Names the log file to display. If a simple filename is provided, with no slashes, the file is assumed to reside in *dcelocal***/var/dfs/adm**; the standard choices from that directory are **BakLog**, **BosLog**, **DfsgwLog**, **FlLog**, **FtLog**, **RepLog**, and **UpLog**.

Pathnames are interpreted relative to *dcelocal***/var/dfs/adm**; absolute pathnames are also allowed. In cases where a / (slash) appears in the specified filename, the issuer's username must appear in the **admin.bos** file on the machine specified by the **-server** option.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If the filename specified by **-file** contains a / (slash), the command fails if you use this option and DFS authorization checking is not disabled on the machine specified by -server. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos getlog** command displays the contents of the log file specified by **-file** that resides on the machine specified by **-server** . It can be used to view any of the following log files:

BakLog
Generated by the Backup Server process on each Backup Database machine

BosLog
Generated by the BOS Server process on each server machine

DfsgwLog
Generated by the Gateway Server process on each Gateway Server machine

FlLog Generated by the Fileset Location Server process on each Fileset Database machine

FtLog Generated by the Fileset Server process on each File Server machine

RepLog
Generated by the Replication Server process on each server machine

UpLog
Generated by the **upserver** process on each server machine running the server portion of the Update Server

By default, the command looks in the *dcelocal***/var/dfs/adm** directory for the log file it is to display. It is not necessary to specify the full pathname of a log file if it resides in the default directory. However, if the file resides elsewhere, the full pathname of the log file must be provided. (The command can also be used to view the **.old** version of a log file created by the associated server process.)

Privilege Required

No privilege is required if the filename specified by **-file** does not contain a / (slash). If the name contains a / (slash), the issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following example displays the contents of the **BosLog** file located in the default directory (*dcelocal***/var/dfs/adm**) on the server machine named **fs1**:

```
$ bos getl /.../abc.com/hosts/fs1 BosLog
```

RELATED INFORMATION

Files: **BakLog(4dfs)**, **BosLog(4dfs)**, **DfsgwLog(4dfs)**, **FlLog(4dfs)**, **FtLog(4dfs)**, **RepLog(4dfs)**, **UpLog(4dfs)**.

bos getrestart

Purpose

Lists automatic restart times for server processes

Synopsis

bos getrestart -server *machine*[**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine on which to check the restart times. The BOS Server on this machine executes the command. Specify the machine's DCE pathname, its host name, or its IP address.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos getrestart** command displays the following two restart times from the **BosConfig** file on the server machine specified by the **-server** option:

- The general restart time, which is the time each week when the BOS Server process automatically restarts itself and all processes that have the status flag **Run** in the **BosConfig** file
- The new binary restart time, which is the time each day when the BOS Server automatically restarts any process executed from a binary file in the *dcelocal***/bin** directory whose time stamp is later than the last restart time for the process

Either of these times can be daily (consist only of a time) or weekly (consist of a day and time). By default, the general restart time is once a week, while the new binary restart time occurs once a day. Both restart times are set with the **bos setrestart** command.

Privilege Required

No privileges are required.

OUTPUT

The output consists of the following two lines:

Server *machine* restarts at *time* Server *machine* restarts for new binaries at *time*

where *machine* indicates the name of the server machine whose restart times are displayed, and possible values for *time* include the following:

never Indicates that the BOS Server never performs that type of restart

A specified day and time
Indicates that the BOS Server performs that type of restart once per week

A specified time
Indicates that the BOS Server performs that type of restart once per day

EXAMPLES

The following command displays the restart times for the server machine **fs2**:

```
$ bos getr /.../abc.com/hosts/fs2


Server fs2 restarts at sun 4:00 am
Server fs2 restarts for new binaries at 2:15 am
```

RELATED INFORMATION

Commands: **bos setrestart(8dfs)**.

Files: **BosConfig(4dfs)**.

bos help

Purpose

Shows syntax of specified **bos** commands or lists functional descriptions of all **bos** commands

Synopsis

bos help[**-topic** *string*][**-help**]

OPTIONS

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **status**, not **bos status**). If this option is omitted, the output provides a short description of all **bos** commands.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos help** command displays the first line (name and short description) of the online help entry for every **bos** command if -topic is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **bos apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

OUTPUT

The online help entry for each **bos** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with **Usage:**, lists the command options in the prescribed order.

EXAMPLES

The following command displays the online help entry for the **bos status** command:

```
$ bos help status


bos status: show server process status
Usage: bos status -server <machine> [-process <server_process>...]
[-long] [-noauth | -localauth] [-help]
```

RELATED INFORMATION

Commands: **bos apropos(8dfs)**.

bos install

Purpose

Installs new versions of binary files

Synopsis

bos install -server *machine* **-file** *binary_file* [**-dir** *alternate_dest*] [**-noauth** | **-localauth**] [**-help**]

OPTIONS

-server *machine*
Names the server machine on which the new binary files are to be installed. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-file *binary_file*
Specifies the pathname of each binary file to be installed on the machine specified by the **-server** option. For each file, specify either the full pathname or a relative pathname (one that does not begin with a slash); relative pathnames are interpreted relative to the current working directory. The name of each file remains the same when it is installed on the specified machine; the command automatically preserves an existing file of the same name as a file that is installed.

-dir *alternate_dest*
Provides the pathname of the directory on the machine specified by the -server option in which all specified files are to be installed. Omit this option to install the files in the default directory, *dcelocal***bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on the machine specified by **-server**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos install** command installs each binary file specified with the **-file** option on the server machine specified with the **-server** option. The **-file** option provides the

pathname of each file to be installed on the specified machine. By default, the command installs the files in the *dcelocal*/**bin** directory on the specified machine; use the **-dir** option to indicate a different installation directory on the specified machine.

The command does not change the names of files when it installs them. To preserve the current version of a binary file that has the same name as a file being installed, the command adds a **.BAK** extension to the name of the existing file. If there is a **.BAK** version at least 7 days old, the command adds a **.OLD** extension to its name and it replaces the current **.OLD** version (if one exists). If there is a **.BAK** version less than 7 days old, it is overwritten when the current version receives a **.BAK** extension. If there is no **.OLD** version, the current **.BAK** version becomes the **.OLD** version automatically, regardless of its age.

The command is typically used to install new versions of binary files on Binary Distribution machines. The machine specified with the **-server** option should be the Binary Distribution machine for its CPU/operating system type. If it is not, newly installed binary files are overwritten the next time the **upclient** process on the specified machine copies new (or different) versions of binary files via the **upserver** process on the Binary Distribution machine of its CPU/operating system type. (Note that the Update Server propagates binary files from Binary Distribution machines, but the BOS Server installs files when the **bos install** command is issued; by default, it takes the Update Server 5 minutes to propagate binary files from a Binary Distribution machine.)

To make the machine specified by **-server** immediately start using new binary files for server processes controlled by the BOS Server, issue the **bos restart** command. Otherwise, new binaries are not used until the BOS Server restarts the affected processes at the new binary restart time specified in the *dcelocal*/**var/dfs/BosConfig** file. Use the **bos getrestart** and **bos setrestart** commands to inspect and set the new binary restart time. (The information in this paragraph applies *only* to affected processes already under the control of the BOS Server.)

The **bos install** command installs all files with the UNIX mode bits set to **755** (**rwxr−xr−x**), regardless of the mode bits associated with a version of the file that currently exists in the destination directory. These permissions are subject to the **umask** associated with the BOS Server on the machine on which the files are installed (because the BOS Server on the specified machine actually executes the command). The mode bits associated with the current version of the file are preserved when it becomes the **.BAK** version, as are those of the **.BAK** version when it becomes the **.OLD** version. (The command does not preserve the access control list, or ACL, permissions of a file installed from a DCE LFS fileset, nor does it directly manipulate the ACL permissions of a file installed into a DCE LFS fileset.)

Use the **bos uninstall** command to replace the current version of a binary file with the next-oldest version of the file: the **.BAK** version, if it exists; otherwise, the **.OLD** version. If both the **.BAK** and **.OLD** versions exist, the **.OLD** version replaces the **.BAK** version when the latter becomes the current version. Use the **-all** option with the **bos uninstall** command to remove all versions of a file; use the **bos prune** command to remove **.BAK** and **.OLD** files from the *dcelocal*/**bin** directory. (This command can also be used to remove core files from the *dcelocal*/**var/dfs/adm** directory.) Use the **bos getdates** command to check the time stamps on binary files.

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**.

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos getdates(8dfs)**, **bos getrestart(8dfs)**, **bos prune(8dfs)**, **bos restart(8dfs)**, **bos setrestart(8dfs)**, **bos uninstall(8dfs)**, **upclient(8dfs)**, **upserver(8dfs)**.

Files: **BosConfig(4dfs)**.

bos lsadmin

Purpose

Lists the users, groups, and servers from an administrative list

Synopsis

bos lsadmin -server *machine* **-adminlist** *filename* [**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine that houses the administrative list whose principals and groups are to be displayed. The BOS Server on this machine executes the command. Specify the machine's DCE pathname, its host name, or its IP address.

-adminlist *filename*
Names the administrative list whose principals and groups are to be displayed. The complete pathname is unnecessary if the list is stored in the default configuration directory (*dcelocal***/var/dfs**).

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos lsadmin** command lists the principal names of users and server machines and the names of groups found in the administrative list specified by the **-adminlist** option on the server machine specified by the **-server** option. Principals whose names are specified in the administrative list or that are members of groups specified in the list can issue administrative commands for the DFS server process associated with the list.

The default path for the administrative lists is the configuration directory (*dcelocal***/var/dfs**). If the specified list is stored in the default directory, only the specific filename is required. If the specified list is stored elsewhere, the pathname to the file that was used when the associated server process was started is required.

Use the **bos addadmin** command to add principals and groups to an administrative list. Use the **bos rmadmin** command to remove principals and groups from an administrative list.

Privilege Required

No privileges are required.

OUTPUT

The command displays the output

```
Admin Users are:
```

followed by the principal name of each user and machine and the name of each group contained in the administrative list. Names from the local cell are displayed in an abbreviated form (for example, *username* for */.../cellname/ username*); names from foreign cells are displayed in full. Each name is preceded by one of the following strings:

user: Precedes the principal name of each user or machine from the local cell

foreign_user:
Precedes the principal name of each user or machine from a foreign cell

group:
Precedes the name of each group from the local cell

foreign_group:
Precedes the name of each group from a foreign cell

EXAMPLES

The following command lists the members of the **admin.bos** file on the server machine named **fs1**. The administrative list contains two users, a server machine, and two groups, all of which are from the local cell.

```
$ bos lsa -server /.../abc.com/hosts/fs1 -adminlist admin.bos


Admin Users are: user: jones, user: smith,
user: hosts/fs1/self, group: dfs-admin, group: fs1-admin
```

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos rmadmin(8dfs)**.

Files: **admin.bak(4dfs)**, **admin.bos(4dfs)**, **admin.fl(4dfs)**, **admin.ft(4dfs)**, **admin.up(4dfs)**.

bos lscell

Purpose

Lists the cell in which the BOS Server is running

Synopsis

bos lscell -server *machine* [**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine on which the BOS Server whose cell is to be listed is running. Specify the machine's DCE pathname, its host name, or its IP address.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos lscell** command reports the name of the cell in which the BOS Server on the machine specified with the **-server** option is running. The command extracts the information from the local configuration file, *dcelocal***/dce_cf.db**, on the specified machine. If the command fails after being issued from the machine specified by **-server** (if **-server** is the local machine), the failure may indicate that the local **dce_cf.db** file is corrupted; use the **cat** or **more** command (or a similar command appropriate to your operating system) to display the contents of the file, and ensure that it is not corrupted.

Privilege Required

No privileges are required.

OUTPUT

The **bos lscell** command displays the following line reporting the name of the cell in which the BOS Server is running:

Cell name is *cellname*

EXAMPLES

The following command displays the name of the cell in which the BOS Server on the machine named **fs1** is running:

```
$ bos lscell /.../abc.com/hosts/fs1
```

```
Cell name is abc.com
```

bos lskeys

Purpose

Displays server encryption key information from a keytab file

Synopsis

bos lskeys -server *machine*[**-principal** *name*][**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine whose keytab file is to have keys listed. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-principal *name*
Provides the principal name for which associated keys are to be listed. The default is the DFS principal name of the machine specified by **-server**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos lskeys** command formats and displays information about server encryption keys kept in the **/krb5/v5srvtab** keytab file on the server machine specified by **-server**. It displays information for keys associated with the principal name indicated by **-principal** ; the DFS principal name of the server machine specified with **-server** is used by default.

DFS authorization checking must be disabled on the machine specified with -server to display the string of octal numbers that compose the key. (Use the **bos setauth** command to disable DFS authorization checking.) Disabling DFS authorization checking is required for two reasons. First, it implies that only someone authorized to issue the **bos setauth** command or someone with **root** access to **-server** 's local disk (presumably a system administrator) is able to see actual encryption keys. Second, it makes it clear that the system is in a compromised state of security while

server encryption keys are being examined. (Both turning off DFS authorization checking and displaying keys on a screen are serious security risks.)

If DFS authorization checking is enabled on **-server** (the normal case), a **checksum** appears in place of the octal numbers. A checksum is a decimal number derived by encrypting a constant with each key.

Privilege Required

If DFS authorization checking is enabled, you must be listed in the **admin.bos** file on the machine specified by **-server** ; checksums are displayed instead of the actual keys. Because DFS authorization checking must be disabled with the **bos setauth** command before the actual keys (rather than just checksums) can be displayed, no privilege is required to see the keys. However, you must be listed in the **admin.bos** file on a machine to use the **bos setauth** command to disable DFS authorization checking on it.

OUTPUT

The **bos lskeys** command displays one line for each server encryption key associated with **-principal** in the keytab file on the machine specified by **-server**. Each key is identified by its key version number. If DFS authorization checking is enabled on the machine, a checksum is displayed with each version number; if checking is disabled, the octal numbers that constitute the key are displayed.

A line specifying when the key in the Registry Database (at the Registry Server) was last changed follows the list of keys or checksums. The words **All done** indicate the end of the output.

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS Server is using the packet integrity protection level instead:

Data encryption unsupported by RPC. Continuing without it.

EXAMPLES

The following command shows the checksums for the keys associated with the principal name of **fs3** in the keytab file on that machine. The checksums appear instead of the actual keys because DFS authorization checking is *not* disabled.

```
$ bos lsk /.../abc.com/hosts/fs3

key 1 has cksum 972037177
key 3 has cksum 282517022
key 4 has cksum 260617746
Keys last changed (at the registry server) on Thu Jun 6 11:24:46 1991.
All done.
```

The following command lists the keys associated with **fs3** after DFS authorization checking is disabled with the **bos setauth** command:

```
$ bos setauth /.../abc.com/hosts/fs3 off

$ bos lsk /.../abc.com/hosts/fs3

key 1 is'\040\205\211\241\345\002\023\211'
key 2 is'\343\315\307\227\255\320\135\244'
key 3 is'\310\310\255\253\265\236\261\211'
Keys last changed (at the registry server) on Thu Jun 6 11:24:46 1991.
All done.
```

RELATED INFORMATION

Commands: **bos addkey(8dfs)**, **bos gckeys(8dfs)**, **bos genkey(8dfs)**, **bos rmkey(8dfs)**, **bos setauth(8dfs)**, **keytab(8dce)**.

Files: **v5srvtab(5sec)**.

bos prune

Purpose

Removes old binary and core files from *dcelocal*/**bin** and *dcelocal*/**var/dfs/adm**

Synopsis

bos prune -server *machine*[**-bak][-old][-core][-all**][**-noauth** | **-localauth**] [**-help**]

OPTIONS

-server *machine*
Names the server machine from which to remove the indicated files. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-bak Removes all files with a **.BAK** extension from *dcelocal*/**bin**. Use this option with **-old**, **-core**, or both, or use -all.

-old Removes all files with an.**OLD** extension from *dcelocal*/**bin**. Use this option with **-bak**, **-core**, or both, or use -all.

-core Removes all core files from *dcelocal*/**var/dfs/adm**. Use this option with **-bak**, **-old**, or both, or use **-all**.

-all Removes all **.BAK** and **.OLD** files from *dcelocal*/**bin** and all core files from *dcelocal*/**var/dfs/adm**. Use this option or use some combination of **-bak**, **-old**, and **-core**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos prune** command removes obsolete versions of binary and core files from the *dcelocal*/**bin** and *dcelocal*/**var/dfs/adm** directories on the server machine specified with the **-server** option. Binary files should only need to be removed from the Binary Distribution machine for a CPU/operating system type; core files may need to be removed from any server machine. Specify the files to be removed with the command's other options as follows:

- Use the **-bak** option to remove all **.BAK** files from *dcelocal***/bin**.
- Use the **-old** option to remove all **.OLD** files from *dcelocal***/bin**.
- Use the **-core** option to remove all core files from *dcelocal***/var/dfs/adm**.
- Use the **-all** option to remove all three types of files.

The **-bak**, **-old**, and **-core** options can be combined to remove different types of files with the same command. The **-all** option can also be used with any of the three options, but using the **-all** option alone is sufficient to remove all three types of files.

Binary files with **.BAK** and **.OLD** extensions are created when new versions of binary files are installed with the **bos install** command. Core files are created when a process that the BOS Server is monitoring goes down.

Use the **bos uninstall** command to replace the current version of a binary file with its next-oldest version (**.BAK** or, if the **.BAK** version does not exist, **.OLD**) or to remove all versions of a binary file. Use the **bos getdates** command to determine the time stamps on binary files.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

RELATED INFORMATION

Commands: **bos getdates(8dfs)**, **bos install(8dfs)**, **bos uninstall(8dfs)**.

bos restart

Purpose

Restarts processes on a server machine

Synopsis

bos restart -server *machine*[**-bosserver** | **-process** *server_process*][**-help**]

OPTIONS

-server *machine*
Names the server machine on which to stop and restart indicated processes. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-bosserver
Indicates that all processes, including the current BOS Server, are to stop running. A new BOS Server immediately starts; it then restarts all processes with the status flag **Run** in the *dcelocal***/var/dfs/BosConfig** file.

Provide this option or provide the **-process** option. Omit both options to stop all processes except the BOS Server; those with the status flag **Run** in the **BosConfig** file are immediately restarted.

-process *server_process*
Specifies each process to be stopped and immediately restarted. The BOS Server stops all specified processes that are currently running; it then restarts all of the specified processes, regardless of their status flags in the **BosConfig** file. Refer to a process by the name assigned with the **-process** option of the **bos create** command (this name appears in the output from the **bos status** command). *Do not include***bosserver** *in the list of processes*; use the -bosserver option instead.

Provide this option or provide the **-bosserver** option. Omit both options to stop all processes except the BOS Server; those with the status flag **Run** in the **BosConfig** file are immediately restarted.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos restart** command instructs the BOS Server running on the server machine specified by **-server** to stop all indicated processes on the machine. The BOS Server then immediately restarts some or all of the processes, depending on the options included with the command. The processes to be stopped and possibly restarted are specified with the following options:

1. The **-bosserver** option causes the BOS Server to stop all processes, including itself. A new BOS Server immediately starts; it then restarts all processes with the status flag **Run** in the **BosConfig** file.
2. The **-process** option causes the BOS Server to stop and immediately restart all specified processes, regardless of their status flags in the **BosConfig** file. All restarted processes with the status flag **NotRun** in the **BosConfig** file have the status **temporarily enabled** in the output of the **bos status** command.
3. The absence of both the **-bosserver** and **-process** options causes the BOS Server to stop all processes except itself. The BOS Server then immediately restarts all processes with the status flag **Run** in the **BosConfig** file.

This command can be used to stop only those processes the BOS Server controls. Also, it does *not* change the status flag for a process in the **BosConfig** file.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command instructs the BOS Server on */.../abc.com/hosts/fs3* to stop all processes, including itself. A new BOS Server immediately starts, after which it restarts all processes with the status flag **Run** in the **BosConfig** file.

```
$ bos restart /.../abc.com/hosts/fs3 -bos
```

The following command instructs the BOS Server on */.../abc.com/hosts/fs5* to stop all processes except itself. The BOS Server then restarts all processes with the status flag **Run** in the **BosConfig** file.

```
$ bos res /.../abc.com/hosts/fs5
```

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos status(8dfs)**.

Files: **BosConfig(4dfs)**.

bos rmadmin

Purpose

Removes a user, group, or server from an administrative list

Synopsis

bos rmadmin -server *machine***-adminlist** *filename*[**-principal** *name*][**-group** *name*][**-removelist**][**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine that houses the administrative list from which principals, groups, or both are to be removed. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-adminlist *filename*
Names the administrative list from which principals, groups, or both are to be removed. The complete pathname is unnecessary if the list is stored in the default configuration directory (*/dcelocal***/var/dfs**).

-principal *name*
Specifies the principal name of each user or server machine to be removed from the administrative list. A user from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** */cellname/ username* or just *username*); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, ***/.../*** */cellname/***hosts/** */hostname/***self** or just **hosts/** */hostname/***self**); a server machine from a foreign cell can be specified only by a full principal name.

-group *name*
Specifies the name of each group to be removed from the administrative list. A group from the local cell can be specified by a full or abbreviated group name (***/.../*** */cellname/ group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

-removelist
Specifies that the file indicated with **-adminlist** is to be removed if it is empty either when the command is issued or after any principals or groups specified with the command are removed. This option has no effect if the specified file is not empty when the command is issued or after any indicated principals or groups are removed.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if

the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos rmadmin** command removes the specified users, groups, and servers from the administrative list specified by the **-adminlist** option on the server machine specified by the **-server** option. The principal (login) names of users and the principal names of server machines to be removed from the administrative list are specified with the **-principal** option; the names of groups to be removed from the list are specified with the **-group** option. Principals removed from the administrative list either directly (with the -principal option) or indirectly (as members of groups indicated with the -group option) can no longer issue administrative commands for the DFS server process associated with the list.

The default path for administrative lists is the configuration directory (*dcelocal*/**var/dfs**). If the specified list is stored in the default directory, only the specific filename is required. If the specified list is stored elsewhere, the pathname to the file that was used when the associated server process was started is required.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command removes the former administrative users smith and jones from the **admin.bos** file on **fs1**:

```
$ bos rmadmin -server /.../abc.com/hosts/fs1 -adminlist admin.bos\
-principal smith jones
```

RELATED INFORMATION

Commands: **bos addadmin(8dfs)**, **bos lsadmin(8dfs)**.

Files: **admin.bak(4dfs)**, **admin.bos(4dfs)**, **admin.fl(4dfs)**, **admin.ft(4dfs)**, **admin.up(4dfs)**.

bos rmkey

Purpose

Removes server encryption keys from a keytab file

Synopsis

bos rmkey -server *machine***-kvno** *version_number*[**-principal** *name*]
[**-noauth** | **-localauth**][**-help**]

OPTIONS

-server *machine*
Names the server machine whose keytab file is to have keys removed from it. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-kvno *version_number*
Specifies the key version number of each key to be removed from the keytab file. The command removes each key that is associated with a specified key version number and the principal indicated by **-principal**. Each version number must be an integer in the range 1 to 255.

-principal *name*
Provides the principal name associated with the keys to be removed from the keytab file. The default is the DFS principal name of the machine specified by -server.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos rmkey** command removes server encryption keys from the **/krb5/v5srvtab** keytab file on the server machine specified by **-server**. It removes each key associated with a key version number indicated by **-kvno** and the principal indicated by **-principal**. The command has no effect on the Registry Database. Once a key is removed from the keytab file, it can no longer be used to establish communication between clients and the server to which it applied.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

OUTPUT

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS Server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

EXAMPLES

The following command removes two keys from the keytab file on **fs1**: the keys with key version numbers **5** and **6** that are associated with the DFS principal name of **fs1**.

```
$ bos rmk /.../abc.com/hosts/fs1 -kvno 5 6
```

RELATED INFORMATION

Commands: **bos addkey(8dfs)**, **bos gckeys(8dfs)**, **bos genkey(8dfs)**, **bos lskeys(8dfs)**, **keytab(8dce)**.

Files: **v5srvtab(5sec)**.

bos setauth

Purpose

Enables or disables DFS authorization checking for all DFS server processes on a machine

Synopsis

bos setauth -server *machine***-authchecking** {**onoff**}[**-noauth** | **-localauth**] [**-help**]

OPTIONS

-server *machine*
Names the server machine on which the status of DFS authorization checking is to change. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-authchecking
Determines whether or not server processes on the machine check for DFS authorization. A value of **on** enables DFS authorization checking; a value of **off** disables it.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. (The option can be used only when enabling authorization checking.) If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos setauth** command enables or disables DFS authorization checking on the server machine specified by the **-server** option. If DFS authorization checking is enabled on a server machine, all DFS server processes running on the machine check that the issuer of a command is correctly authorized (is included in the necessary administrative lists) to execute the command. If DFS authorization checking is disabled on a server machine, the DFS server processes on the machine perform any action for any user, even the unprivileged user **nobody**.

By default, DFS authorization checking is enabled on every server machine. Disabling it on a server machine is a serious security risk. It is typically disabled for the briefest possible time and only in the following situations:

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file

To indicate to all DFS server processes (including itself) that DFS authorization checking is disabled on a server machine, the BOS Server creates the zero-length file */dcelocal***/var/dfs/NoAuth** on the local disk of the machine. All DFS server processes, including the BOS Server, check for the presence of this file when they are requested to perform an operation; they do not check for the necessary administrative privilege for a requested operation when the file is present. To indicate that DFS authorization checking is enabled, the BOS Server removes the file.

Enter this command with the **-authchecking** option and an argument of **off** to disable DFS authorization checking on a server machine. (DFS authorization checking can also be disabled by including the **-noauth** option with the **bosserver** command used to start the BOS Server.) Issue the command with the **-authchecking** option and an argument of **on** to enable DFS authorization checking on a server machine. It is not necessary to restart currently running server processes when you change the state of DFS authorization checking; server processes immediately obey the current state of DFS authorization checking and act accordingly.

The **bos status** command can be used to determine whether DFS authorization checking is enabled or disabled on a server machine. The command displays the following message if DFS authorization checking is disabled on a machine. (It does not display the message if DFS authorization checking is enabled.)

```
Bosserver reports machine is not checking authorization.
```

The **-noauth** option available with many **bos** and **fts** commands is used when authentication information is unnecessary or unavailable. Use the **-noauth** option if DFS authorization checking is disabled on a server machine on which administrative privilege is required or if the Security Service is unavailable.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server** to disable DFS authorization checking on that machine. (No privilege is required to enable DFS authorization checking if it is currently disabled.)

CAUTIONS

Always use the **bos setauth** command to create the */dcelocal***/var/dfs/NoAuth** file. Do not create the file directly except when explicitly told to do so by instructions for dealing with emergencies (such as emergencies involving server encryption keys). Creating the file directly requires logging into the local operating system of a machine as **root** and using the **touch** command (or its equivalent).

EXAMPLES

The following command disables DFS authorization checking for all DFS server processes on the server machine named **fs7**:

```
$ bos seta /.../abc.com/hosts/fs7 off
```

RELATED INFORMATION

Commands: **bos status**, **bosserver(8dfs)**.

Files: **NoAuth(4dfs)**.

bos setrestart

Purpose

Sets the date and time at which the BOS Server restarts all processes or only those with new binaries

Synopsis

```
bos setrestart -server machine{-general time -newbinary time}
[-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*

Specifies the server machine for which restart times are to be set. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-general *time*

Sets the time at which the BOS Server restarts first itself and then each server process that has an entry in the **BosConfig** file with a status flag of **Run**. Specify a day and time to perform the restart weekly at that time; specify a time to perform the restart daily at that time. Day and time specifications have the following format:

*day***]** *hh***:***mm*

Enter the name of the day in all lowercase letters, giving either the whole name or the first three letters as an abbreviation (for example, **sunday** or **sun**). Specify the time of day by separating the hours from the minutes with a **:** (colon). Use 24-hour time (for example, **14:30**), or use 1:00 through 12:00 with **am** or **pm** (for example, ″**2:30 pm**″). As shown in the example, enclose the entry in ″ ″ (double quotes) if it contains a space.

You can use either of two additional specifications instead of a time or a day and time:

never Directs the BOS Server never to perform the indicated type of restart

now Directs the BOS Server to use the day and time at which the command is issued (for example, Sunday at 2:00 a.m.) as the day and time for the indicated type of restart

If this option is never used to set the general restart time, the default general restart time is Sunday at 4:00 a.m. If you change the general restart time, the recommended frequency for this type of restart is once per week.

-newbinary *time*

Sets the time at which the BOS Server restarts any server process whose binary file was installed in *dcelocal***/bin** after the current instance of the process started running. The recommended frequency for this type of restart is once per day, so it is standard to specify only a time of day. Use

the conventions described for the **-general** option to express the time of day. The remarks for the **-general** option concerning **never** and **now** also apply to this option.

If this option is never used to set the binary checking time, the default binary checking time is 5:00 a.m.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos setrestart** command sets the times at which the BOS Server running on the server machine specified by the **-server** option is to perform one of two types of restarts. The command records the time settings in the **BosConfig** file. The two types of restart times are

- The time each week when the BOS Server restarts itself and any processes marked with the status flag **Run** in the **BosConfig** file. This is equivalent to executing **bos restart** with the **-bosserver** option. The default setting is 4:00 a.m. each Sunday morning.
- The time each day when the BOS Server restarts any process currently running for which the binary file in *dcelocal***/bin** was modified since the process was last started (or restarted). The default setting is 5:00 a.m. each day.

To change both times, you must issue the command twice.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by the **-server** option.

CAUTIONS

Restarting processes makes them unavailable for a period of time. It is advisable to set the restarts for times of typically low usage to inconvenience as few users as possible.

If the specified time is within one hour of the current time, the BOS Server does not restart the processes until that time on the next day.

EXAMPLES

The following command defines a general restart time in the **BosConfig** file on **fs4** that causes all processes on that machine to stop and restart each Saturday morning at 3:30 a.m.:

```
$ bos setr -s /.../abc.com/hosts/fs4 -gen "sat 3:30"
```

The following command defines a new binary restart time in the **BosConfig** file on **fs6**, instructing the BOS Server on that machine to check for new binary files each evening at 11:45 p.m. and restart any processes for which it finds a new file at that time:

```
$ bos setr -s /.../abc.com/hosts/fs6 -new 23:45
```

RELATED INFORMATION

Commands: **bos getrestart(8dfs)**, **bos restart(8dfs)**.

Files: **BosConfig(4dfs)**.

bos shutdown

Purpose

Stops processes without changing their status flags in the **BosConfig** file

Synopsis

```
bos shutdown -server  machine[-process  server_process][-wait ]
[-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*
: Names the server machine on which the indicated processes are to be stopped. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-process *server_process*
: Specifies each process to be stopped. If this option is omitted, the BOS Server stops all server processes other than itself on the server machine. Refer to a process by the name assigned with the **-process** option of the **bos create** command; this name appears in the output of the **bos status** command.

-wait
: Indicates that the command shell prompt is not to return until the shutdown is complete (until all processes actually stop running). If this option is omitted, the prompt returns almost immediately, even if all of the processes are not yet stopped.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos shutdown** command instructs the BOS Server running on the server machine specified by **-server** to stop either all processes (except itself) running on the machine *or* only the processes specified by -process. The command does not change a process's status flag in the **BosConfig** file, only in the BOS Server's memory state.

Processes stopped with this command do not run again until they are started using the **bos start**, **bos startup**, or **bos restart** commands, or until the BOS Server itself restarts.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command instructs the BOS Server running on **fs3** to stop running all processes except itself:

```
$ bos shutdown -s /.../abc.com/hosts/fs3
```

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos status(8dfs)**.

bos start

Purpose

Starts processes after setting their status flags to **Run** in the **BosConfig** file and in memory

Synopsis

```
bos start -server machine-process server_process[-noauth  | -localauth ]
[-help ]
```

OPTIONS

-server *machine*
: Names the server machine whose processes are to be started. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-process *server_process*
: Specifies each process to be started after its status flag in the **BosConfig** file and in memory is set to **Run**. Refer to a process by the name assigned with the **-process** option of the **bos create** command; this name appears in the output from the **bos status** command.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos start** command changes the status flag for each server process specified by **-process** from **NotRun** to **Run** in the **BosConfig** file and in memory on the server machine specified by **-server**. It then starts each specified process running on that machine.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

CAUTIONS

If an instance of a process is already running, the only effect is to guarantee that its status flag is set to **Run** in both the **BosConfig** file and memory; it does not start a new instance of the process. Issue the **bos restart** command after this command to start a new instance.

EXAMPLES

The following command causes the BOS Server on **fs3** to start the Replication Server (**repserver** process) on that machine by changing its status flags to **Run** in both the **BosConfig** file and memory:

```
$ bos start /.../abc.com/hosts/fs3 repserver
```

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos restart(8dfs)**, **bos startup(8dfs)**, **bos status(8dfs)**.

Files: **BosConfig(4dfs)**.

bos startup

Purpose

Starts processes by changing their status flags to **Run** in memory without changing their status flags in the **BosConfig** file

Synopsis

```
bos startup -server machine[-process server_process]
[-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*
: Names the server machine whose processes are to be started. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-process *server_process*
: Specifies each process to be started after its status flag in memory is set to **Run**. Refer to a process by the name assigned with the -process option of the **bos create** command; this name appears in the output from the **bos status** command.

  If this option is omitted, all server processes with a status flag of **Run** in the **BosConfig** file that are not running are started after their status flags in memory are set to **Run**.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

This command instructs the BOS Server running on the server machine specified by **-server** to start *either* all server processes with a status flag of **Run** in the **BosConfig** file that are not running (if -process is omitted) *or* each process specified by **-process**, even if its status flag in the **BosConfig** file is **NotRun**. The status flags of all started processes are changed from **NotRun** to **Run** in memory.

Using **-process** is useful for testing server processes without enabling them permanently. This command does *not* change the status flag for a process in the **BosConfig** file.

CAUTIONS

If an instance of a process is already running, the only effect is to guarantee that its status flag is set to **Run** in memory; it does not start a new instance of the process. Issue the **bos restart** command after this command to start a new instance.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

EXAMPLES

The following command causes the BOS Server on **fs3** to start all processes on that machine marked with a status flag of **Run** in the **BosConfig** file that are not currently running. The status flags of all such processes are set to **Run** in memory; their status flags remain set to **Run** in the **BosConfig** file.

```
$ bos startup /.../abc.com/hosts/fs3
```

The following command causes the BOS Server on **fs3** to start the Replication Server (**repserver** process) on that machine by changing its status flag to **Run** in memory. The process's status flag remains unchanged in the **BosConfig** file, regardless of its current setting (**Run** or **NotRun**).

```
$ bos startup /.../abc.com/hosts/fs3 repserver
```

RELATED INFORMATION

Command: **bos create(8dfs)**, **bos restart(8dfs)**, **bos shutdown(8dfs)**, **bos start(8dfs)**, **bos status(8dfs)**, **bos stop(8dfs)**.

Files: **BosConfig(4dfs)**.

bos status

Purpose

Displays the statuses of server processes on a server machine

Synopsis

```
bos status -server machine[-process server_process][-long ]
[-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*
: Names the server machine about whose processes status information is to be displayed. The BOS Server on this machine executes the command. Specify the the machine's DCE pathname, its host name, or its IP address.

-process *server_process*
: Specifies each process whose status is to be displayed; refer to a process by the name assigned with the **-process** option of the **bos create** command. If this option is omitted, the statuses of all of the processes on the specified server are listed.

-long
: Directs the BOS Server to provide more detailed information about the specified processes.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos status** command lists status information about the processes on the server machine specified by the **-server** option. Use the **-process** option to indicate the specific processes about which information is to be displayed, or omit the option to display information about all the processes on the server machine. The command also displays appropriate messages if DFS authorization checking is disabled on the machine or if the machine's *dcelocal* directory or a directory or file beneath it has inappropriate protections.

Use the **-long** option to display more information about each specified process. The additional information can be used to determine the role of a server machine in a domain. (See Part 1 of this manual for instructions on using this command to determine the role of a server machine.)

Privilege Required

No privileges are required.

OUTPUT

The command first displays the following line if DFS authorization checking is disabled on the machine. (It does not display the line if DFS authorization checking is enabled.)

```
Bosserver reports machine is not checking authorization.
```

It then displays the following line if the BOS Server finds that the *dcelocal* directory or a directory or file under it on the machine has protections that the BOS Server believes are inappropriate:

```
Bosserver reports inappropriate access on server directories.
```

The message can indicate, for example, that users who should not be able to write to the *dcelocal* directory and its subdirectories have write access. The BOS Server displays the message if the UNIX mode bits on the following objects do not enforce the indicated protections. Provided the mode bits do not violate the specific restrictions cited in the list, a directory or file can grant more permissions than those shown in the list, but it should not grant fewer.

dcelocal
: At least **755**, and **other** cannot have write access

*dcelocal***/bin**
: At least **755**, and **other** cannot have write access

*dcelocal***/var**
: At least **755**, and **other** cannot have write access

*dcelocal***/var/dfs**
: At least **701**, and **other** cannot have write access

*dcelocal***/var/dfs/adm**
: At least **755**, and **other** cannot have write access

*dcelocal***/var/dfs/admin.bos**
: At least **600**, and **other** cannot have write or execute access

The BOS Server also displays the message if all of these objects are not owned by **root**. The BOS Server displays the message only as a courtesy to the user. It does nothing to change the protections on these objects, nor does it fail if these protections are violated.

If the machine's host name or IP address is given for the **-server** option, the **bos** command displays the following message to indicate it is using the unprivileged identity **nobody**:

```
bos: WARNING: short name for server used; no authentication information will be
sent to the bosserver.
```

To run **bos status** using a privileged identity, specify the full DCE pathname.

The command then displays a separate entry for each specified process. The first line of an entry shows the current status of the process. The possible statuses for any process include the following:

currently running normally
For a **simple** process, this means it is currently running; for a **cron** process, this means it is scheduled to run.

temporarily enabled
The status flag for the process in the *dcelocal*/**var/dfs/BosConfig** file is **NotRun**, but the process has been enabled with the **bos startup** or **bos restart** command.

temporarily disabled
Either the **bos shutdown** command was used to stop the process, or the BOS Server quit trying to restart the process, in which case the message **stopped for too many errors** also appears.

disabled
The status flag for the process in the **BosConfig** file is **NotRun**, and the process has not been enabled.

has core file
The process failed or produced a core file at some time. This message can appear with any of the other messages. Core files are stored in *dcelocal*/**var/dfs/adm**. The name of the core file indicates the process that failed (for example, **core.ftserver**).

The output for a **cron** process includes an auxiliary status message that reports when the command is next scheduled to execute.

The command displays the following additional information when the **-long** option is used:

- The process type (**simple** or **cron**).
- How many **proc starts** occurred (**proc starts** occur when the process is started or restarted by the current BOS Server).
- The time of the last **proc start**.
- The exit time and error exit time when the process last failed. This appears only if the process failed while the BOS Server was running. (Provided the BOS Server was running both when the process was started and when it failed, the BOS Server can provide this information for any process that has an entry in the **BosConfig** file.)
- The command and its options used by the BOS Server to start the process.

EXAMPLES

The following command displays the statuses of all server processes on the File Server machine named **fs4**:

```
$ bos status /.../abc.com/hosts/fs4


Instance ftserver, currently running normally.
Instance repserver, currently running normally.
```

If the **-long** option is included with the command, the following additional information is displayed:

```
Instance ftserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:02 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/ftserver'
```

```
Instance repserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:48 1991 (1 proc starts)
Parameter 1 is '/dcelocal/bin/repserver'
```

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos restart(8dfs)**, **bos shutdown(8dfs)**, **bos start(8dfs)**, **bos startup(8dfs)**, **bos stop(8dfs)**.

Files: **BosConfig(4dfs)**.

bos stop

Purpose

Stops processes after changing their status flags in the **BosConfig** file to **NotRun**

Synopsis

```
bos stop -server machine-process server_process[-wait ]
[ -noauth | -localauth ][-help ]
```

OPTIONS

-server *machine*
: Names the server machine on which to stop the processes. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-process *server_process*
: Specifies each process that the BOS Server is to stop. The BOS Server stops a process after setting its status flag in the **BosConfig** file to **NotRun**. Refer to a process by the name assigned with the -process option of the **bos create** command; this name appears in the output from the **bos status** command.

-wait
: Indicates that the command shell prompt is not to return until all specified processes actually stop running. If this option is omitted, the prompt returns almost immediately, even if all of the processes are not yet stopped.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos stop** command sets the status flag for each server process specified by **-process** to **NotRun** in the **BosConfig** file on the server machine specified by **-server** ; it then stops each process.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

RELATED INFORMATION

Commands: **bos create(8dfs)**, **bos shutdown(8dfs)**, **bos status(8dfs)**.

Files: **BosConfig(4dfs)**.

bos uninstall

Purpose

Installs the former versions of binary files

Synopsis

```
bos uninstall -server machine-file binary_file[-dir alternate_dest][-all ]
[-noauth  | -localauth ][-help ]
```

OPTIONS

-server *machine*
Names the server machine on which the former versions of specified binary files are to be used. The BOS Server on this machine executes the command. To run this command using a privileged identity, specify the full DCE pathname of the machine. To run this command using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option), specify the machine's host name or IP address.

-file *binary_file*
Names each binary file to be replaced with its next-oldest version (**.BAK**, if it exists; otherwise, **.OLD**). All specified files must reside in the same directory (*dcelocal***/bin**, by default, or an alternate directory specified with the **-dir** option). Specify only filenames; if a pathname is provided for a file, the command ignores all but the final element.

-dir *alternate_dest*
Provides the pathname of the directory in which all specified files reside. Omit this option if the files reside in the default directory, *dcelocal***/bin**; otherwise, provide a full or relative pathname. Relative pathnames (pathnames that do not begin with a slash) are interpreted relative to the *dcelocal* directory on the machine specified by **-server**.

-all Directs the BOS Server on the indicated machine to remove all versions (current, **.BAK**, and **.OLD**) of the specified files. Only versions of the files that reside in the *dcelocal***/bin** directory (or an alternate directory specified with the **-dir** option) are removed.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **bos uninstall** command replaces each binary file specified with the **-file** option with its next-oldest version. Use the **-server** option to specify the name of the server machine that houses the files to be manipulated. All specified files must reside in the same directory. By default, the command looks for the files in the *dcelocal***/bin** directory; use the **-dir** option to name a different directory. Versions of the files in other directories on the specified machine are not affected.

The command applies the following algorithm to the removal of each binary file:

- If current, **.BAK**, and **.OLD** versions of the file exist, the command makes the **.BAK** version the current version and it makes the **.OLD** version the **.BAK** version.
- If any version of the file does not exist, the command omits it from the algorithm. For example, if no **.BAK** version exists, the command makes the **.OLD** version the current version.
- If the **-all** option is included with the command, the command removes all versions (current, **.BAK**, and **.OLD**) of the file that exist.
- The command displays the following message if no version of the file exists, or if only the current version exists and the **-all** option is omitted:

  ```
  bos: failed to uninstall filename (there is no earlier
  version present to reinstall)
  ```

  where *filename* is the name of the file that cannot be replaced or removed.

The machine specified with the **-server** option should be the Binary Distribution machine for its CPU/operating system type. If it is not, the binary files are overwritten the next time the **upclient** process on the specified machine copies new (or different) versions of binary files via the **upserver** process on the Binary Distribution machine of its CPU/operating system type. (Note that the Update Server propagates binary files from Binary Distribution machines, but the BOS Server manipulates files when the **bos uninstall** command is issued; by default, it takes the Update Server 5 minutes to propagate binary files from a Binary Distribution machine.)

To make the machine specified by **-server** start using the reinstalled binary files immediately, issue the **bos restart** command. Otherwise, the binaries are not used until the BOS Server restarts the affected process at the new binary restart time specified in the *dcelocal***/var/dfs/BosConfig** file. Use the **bos getrestart** and **bos setrestart** commands to inspect and set the new binary restart time. (The information in this paragraph applies *only* to affected processes already under the control of the BOS Server.)

Use the **bos install** command to install new versions of binary files on a server machine. Use the **bos prune** command to remove **.BAK** and **.OLD** files from the *dcelocal***/bin** directory. (This command can also be used to remove core files from the *dcelocal***/var/dfs/adm** directory.) Use the **bos getdates** command to check the time stamps on binary files.

Privilege Required

The issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

RELATED INFORMATION

Commands: **bos getdates(8dfs)**, **bos getrestart(8dfs)**, **bos install(8dfs)**, **bos prune(8dfs)**, **bos restart(8dfs)**, **bos setrestart(8dfs)**, **upclient(8dfs)**, **upserver(8dfs)**.

Files: **BosConfig(4dfs)**.

bosserver

Purpose

Initializes the Basic OverSeer (BOS) Server process

Synopsis

bosserver[**-adminlist** *filename*][**-noauth**][**-help**]

OPTIONS

-adminlist *filename*
Specifies the file that contains principals and groups authorized to execute **bosserver** RPCs (usually using **bos** commands). If this option is omitted, the **bosserver** obtains the list of authorized users from the default administrative list file, *dcelocal***/var/dfs/admin.bos**.

-noauth
Invokes the **bosserver** process with DFS authorization checking turned off. In this mode, DFS processes, including the **bosserver** process, do not check to see whether issuers have the necessary privilege to enter administrative commands.

This option is intended for use when the BOS Server is initially installed on a server machine. Because it starts the **bosserver** process with DFS authorization checking turned off, it allows the issuer to add members to the **admin.bos** administrative list and to add a key to the keytab file on the server machine.

Use this mode sparingly, as it presents a security risk. Using this option forces all DFS server processes on the machine to run without DFS authorization checking.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **bosserver** command. See the **bos help** and **bos apropos** pages for examples of using these commands.

DESCRIPTION

The Basic OverSeer Server (BOS Server) monitors other DFS server processes, such as the **flserver** and **ftserver** processes, running on the machine and restarts failed processes automatically (without the intervention of a system administrator). The BOS Server, or **bosserver** process, monitors each server process that has a process entry in the local **BosConfig** file. The **bosserver** process must be run on all DFS server machines. The **bosserver** command, which resides in *dcelocal***/bin/bosserver**, is usually added to the proper system start-up file (**/etc/rc** or its equivalent); the process places itself in the background after it starts.

When it is started, the **bosserver** creates the *dcelocal***/var/dfs/adm/BosLog** event log file if the file does not already exist. It then appends messages to the file. If the **BosLog** file exists when the **bosserver** process is started, the process moves it to the **BosLog.old** file in the same directory (overwriting the current **BosLog.old** file if it exists) before creating a new version to which to append messages.

The principals and groups in the **admin.bos** administrative list are authorized to issue BOS commands to stop, start, create, and modify server processes on that machine. For simplified administration, the same **admin.bos** administrative list can be used by all **bosserver** processes in the administrative domain.

The first time the **bosserver** process is initialized, it creates several directories (such as the *dcelocal***/var/dfs/adm** directory and any nonexistent directories along this path), sets the owners to the appropriate identities, and sets the mode bits to provide appropriate access. The **bosserver** process also creates the *dcelocal***/var/dfs/admin.bos** administrative list file and the *dcelocal***/var/dfs/BosConfig** configuration file if either file does not already exist. On subsequent restarts, the process writes the following message to the **BosLog** file if the owners and mode bits of these objects are not set appropriately:

```
Bosserver reports inappropriate access on server directories.
```

See the reference page for the **bos status** command for information about the protections the BOS Server wants to see enforced.

Note: Your vendor can modify the owner of directories created by the BOS Server and the permissions those directories are created with. Refer to your vendor's documentation to determine the protections that apply for your version of DFS.

When initially installing the BOS Server on a server machine, use the **-noauth** option to initialize the **bosserver** process with DFS authorization checking disabled. This creates the **NoAuth** file in the *dcelocal***/var/dfs** directory on the local disk; when the file is present, DFS authorization checking is disabled on the machine.

With DFS authorization checking disabled, add members to the **admin.bos** list and add a key to the keytab file on the server machine. When these steps are complete, use the **bos setauth** command to enable DFS authorization checking. Because running with DFS authorization checking disabled is a serious security risk, enable DFS authorization checking as soon as the previous steps are complete. The **bos status** command can be used to determine whether DFS authorization checking is enabled or disabled on a machine; it displays the following message if DFS authorization checking is disabled on a machine (it does not display the message if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

Privilege Required

The issuer must be logged in as **root** on the local machine.

OUTPUT

If problems are encountered during initialization, the **bosserver** process displays error messages on standard error output. The **bosserver** process keeps an event log in the file *dcelocal***/var/dfs/adm/BosLog**.

RELATED INFORMATION

Commands: **bos setauth(8dfs)**, **bos status(8dfs)**.

Files: **admin.bos(4dfs)**, **BosConfig(4dfs)**, **BosLog(4dfs)**, **NoAuth(4dfs)**.

butc

Purpose

Initializes a Tape Coordinator process

Synopsis

butc[**-tcid** *tc_number*][**-debuglevel** *trace_level*][**-cell** *cellname*][**-help**]

OPTIONS

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) associated with the Tape Coordinator to be initialized. The issuer of **bak** commands uses this number to indicate which Tape Coordinator is to execute a command.

Legal values are the integers from 0 to 1023. The value must match the value assigned to this Tape Coordinator's associated tape drive in the **TapeConfig** file. If this option is omitted, the default is **0** (zero).

-debuglevel *trace_level*
Specifies the kinds of messages the Tape Coordinator produces in its monitoring window. The following two values are legal:

1. A value of **0** (zero) indicates that the Tape Coordinator prompts the issuer only to place new tapes in the drive; the process does not report on its activities (other than to display some output as necessary for operations it executes). This is the default value.
2. A value of **1** indicates that the Tape Coordinator reports on its activities as it restores filesets, in addition to prompting for new tapes as necessary.

-cell *cellname*
Specifies the cell with respect to which the Tape Coordinator is to run. The Tape Coordinator communicates with the Backup Server in the specified cell. The Tape Coordinator can manipulate data in only the specified cell. A host entry must already be defined for the Tape Coordinator machine in the Backup Database of the specified cell.

If this option is omitted, the default is the local cell of the issuer of the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with **butc**. See the **bos help** and **bos apropos** pages for examples using these commands.

-noautoquery
Disables the prompt from the Tape Coordinator to mount the first tape. If the initial prompt is disabled you must either ensure that the tape is mounted before issuing the **bak** command or provide some other method of mounting the tape (such as a user-defined configuration file). See the Guide part of this guide and reference for more information about the user-defined configuration file.

DESCRIPTION

The **butc** command starts a Tape Coordinator process on a Tape Coordinator machine (a machine having a tape drive and an associated Tape Coordinator). The **TapeConfig** file must reside in the directory named */dcelocal/***var/dfs/backup** on the Tape Coordinator machine, and it must contain a single line specifying information about a tape drive and its associated Tape Coordinator if the **butc** process is to start the Tape Coordinator for the drive. A machine to be configured as a Tape Coordinator machine must be a DCE client. Fewer configuration steps are required if the machine is also some type of DFS server machine. (See Part 1 of this manual for complete details about configuring a Tape Coordinator machine.)

The binary file for the **butc** process resides in */dceshared/***bin/butc**. Depending on the operations it executes, the **butc** process that runs as a result of this command contacts the Backup Database (by way of the Backup Server process), the Fileset Location Database (by way of the Fileset Location Server process), or Fileset Server processes.

Enter the **butc** command in a separate terminal session for each Tape Coordinator. (In windowing systems, this generally means a separate window for each Tape Coordinator.) Because the Tape Coordinator must run in the foreground, the terminal session where the **butc** command is issued is unavailable for subsequent commands. Instead, the Tape Coordinator uses it as a dedicated monitoring window on which to display both trace information about filesets it restores and prompts for the insertion of additional tapes into its associated drive. The monitoring window must remain open as long as the Tape Coordinator runs. To stop a Tape Coordinator process, enter an interrupt signal (**<Ctrl-c>** or its equivalent) in the process's monitoring window.

The **butc** program also writes output to two ASCII files in the directory */dcelocal/***var/dfs/backup** on the local disk of the Tape Coordinator machine:

TL_*device_name*
: The **TL**_*device_name* file (where *device_name* is the device name of the tape drive with which the process is associated) is a log file that contains execution information about operations performed by the **butc** process. The level of detail to which each operation is described depends on the operation.

TE_*device_name*
: The **TE**_*device_name* file (where *device_name* is again the device name of the tape drive with which the process is associated) is an error file that contains information about problems encountered by the **butc** process.

The files contain similar information. For example, if you use the **bak dump** command to dump 100 filesets, the log file lists both the names of filesets that were successfully dumped to tape and the names of filesets that, for some reason, were omitted from the dump; the error file, on the other hand, lists the names of only those filesets that were omitted from the dump.

Each time the **butc** process is started for a tape drive and Tape Coordinator pair, it automatically creates the two files. It then appends messages to the files as necessary. If the files already exist when the **butc** process is started, the process moves the current versions to files that end with **.old** extensions (for example, **TL**_*device_name***.old**) before creating new versions of the files to which to append messages. The process overwrites current **.old** files if they exist.

No maintenance is required for the log and error files associated with any tape drive; the files are created automatically each time the **butc** process for a tape drive and Tape Coordinator pair is started. However, the files should be browsed periodically to ensure that operations such as dumps and restores are completing without problems. For example, if a file cannot be dumped because a necessary Fileset Server or Fileset Location Server is unavailable at the time of the dump, the **butc** program writes an appropriate message to the log and error files.

Privilege Required

The issuer must have write and execute permissions on the */dcelocal***/var/dfs/backup** directory, the directory in which the **butc** process creates its log and error files.

RELATED INFORMATION

Commands: **bak(8dfs)**.

Files: **TapeConfig(4dfs)**, **TE(4dfs)**, **TL(4dfs)**.

cm

Purpose

Introduction to the **cm** command suite

OPTIONS

The following options are used with many **cm** commands. They are also listed with the commands that use them.

-path { *filename* **|** *directory_name***}**
Names the files, directories, or both to be used with the command.

-help Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see the **dfs_intro(8dfs)** reference page.

DESCRIPTION

Commands in the **cm** command suite are issued by administrative users to set cache parameters and to update cached information on local workstations. Certain commands in the **cm** suite are available to all users to determine machine and cell information.

The files described in the following sections are used by the Cache Manager to determine its initial configuration and to store and track cached data. Each DFS client machine stores machine-specific versions of these files on its local disk.

The CacheInfo File

The *dcelocal***/etc/CacheInfo** file specifies the Cache Manager's initial configuration. It is manually created during DFS client installation. The Cache Manager checks the file at initialization to determine certain cache configuration information.

The file is a one-line ASCII file that contains three fields separated by colons. The fields provide the following information:

- The local directory where the Cache Manager mounts the DCE global namespace. The default is the global namespace designation (**/...**).
- The local directory to serve as the cache directory. The Cache Manager stores the **CacheItems**, **FilesetItems**, and V files in this directory. The default, *dcelocal***/var/adm/dfs/cache**, can be overridden to direct the Cache Manager to store the files in a different directory.
- The size of the cache in 1024-byte (1-kilobyte) blocks.

The CacheItems File

The *dcelocal***/var/adm/dfs/cache/CacheItems** file is a binary file created and maintained by the Cache Manager. The file records information such as the file ID number and data version number of each V file on a client machine using a disk cache. *Never directly modify or delete this file*; doing so can cause the kernel to panic.

The FilesetItems File

The *dcelocal*/**var/adm/dfs/cache/FilesetItems** file is a binary file created and maintained by the Cache Manager. The file records the fileset-to-mount-point mapping for each fileset accessed by the Cache Manager. The mappings allow the Cache Manager to respond correctly to commands such as **pwd**. *Never directly modify or delete this file*; doing so can cause the kernel to panic.

V Files

The *dcelocal*/**var/adm/dfs/cache/V** *n* files, or V files, hold chunks of cached data on a client machine using a disk cache. In the name of an actual V file, *n* is an integer; each V file has a unique name (for example, **V1**, **V2**, and so on). The format of a V file depends on the information it contains.

By default, each V file holds up to 65,536 bytes (64 kilobytes) of data. The default size can be overridden with the **dfsd** command. *Never directly modify or delete a V file*; doing so can cause the kernel to panic.

Cautions

Specific cautionary information is included with individual commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ man cm
: Displays the reference page for the command suite.

$ man cm_ *command*
: Displays the reference page for an individual command. You must use an _ (underscore) to connect the command suite to the command name. *Do not use the underscore when issuing the command in DFS.*

$ cm help
: Displays a list of commands in a command suite.

$ cm help *command*
: Displays the syntax for a single command.

$ cm apropos -topic *string*
: Displays a short description of any commands that match the specified *string*.

Consult the **dfs_intro(8dfs)** reference page for complete information about the DFS help facilities.

Privilege Required

Specific privileges required by each command are listed with individual commands.

RELATED INFORMATION

Commands: **cm apropos(8dfs)**, **cm checkfilesets(8dfs)**, **cm flush(8dfs)**, **cm flushfileset(8dfs)**, **cm getcachesize(8dfs)**, **cm getdevok(8dfs)**, **cm getsetuid(8dfs)**, **cm help(8dfs)**, **cm lscellinfo(8dfs)**, **cm lsstores(8dfs)**, **cm resetstores(8dfs)**, **cm setcachesize(8dfs)**, **cm setdevok(8dfs)**,

cm setsetuid(8dfs), **cm statservers(8dfs)**, **cm sysname(8dfs)**, **cm whereis(8dfs)**, **dfs_intro(8dfs)**, **dfsd(8dfs)**, **cm getprotectlevels**, **cm setprotectlevels**, **cm getpersistreqs**, **cm setpersistreqs**.

Files: **CacheInfo(4dfs)**, **CacheItems(4dfs)**, **FilesetItems(4dfs)**, **Vn(4dfs)**.

cm apropos

Purpose

Shows each help entry containing a specified string

Synopsis

cm apropos -topic *string* [**-help**]

OPTIONS

-topic *string*
Specifies the keyword string for which to search. If it is more than a single word, surround the string with " " (double quotes) or other delimiters. Type all strings for **cm** commands in lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm apropos** command displays the first line of the online help entry for any **cm** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **cm help** command.

Privilege Required

No privileges are required.

OUTPUT

The first line of an online help entry for a command names the command and briefly describes its function. This command displays the first line for any **cm** command where the string specified by **-topic** is part of the command name or the first line.

EXAMPLES

The following command lists all **cm** commands that have the word **cache** in their names or short descriptions:

```
$ cm apropos cache


flush: flush file data and status information from cache
getcachesize: get cache usage info
setcachesize: set cache size
```

RELATED INFORMATION

Commands: **cm help(8dfs)**.

cm checkfilesets

Purpose

Forces the Cache Manager to update fileset-related information

Synopsis

cm checkfilesets[**-help**]

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **cm checkfilesets** command forces the Cache Manager to discard its table of mappings between fileset names and fileset ID numbers. Because the Cache Manager needs the information in the table to fetch files, this command forces the Cache Manager to fetch the most recent information available about a fileset from the Fileset Location Server before the Cache Manager can fetch any more files from that fileset. (Normally, the Cache Manager flushes the table and constructs a new one every hour.)

This command is most useful if you know that a fileset name has changed or that there is a release of new read-only replicas. Issuing this command forces the Cache Manager to reference the fileset with the new name or the new read-only replica.

To force the Cache Manager to discard a cached file or directory, use the **cm flush** command. To force the Cache Manager to discard any data cached from filesets containing specified files or directories, use the **cm flushfileset** command.

Privilege Required

No privileges are required.

RELATED INFORMATION

Commands: **cm flush(8dfs)**, **cm flushfileset(8dfs)**.

cm flush

Purpose

Forces the Cache Manager to discard data cached from specified files or directories

Synopsis

cm flush[**-path**{ *filename directory_name*}][**-help**]

OPTIONS

-path { *filename* **|** *directory_name***}**
Specifies each file or directory to be flushed. A file for which a full pathname is not specified is assumed to reside in the current working directory. In the case of a directory, all the name mappings and blocks associated with the directory are flushed; data cached from files or subdirectories that reside in the directory is not flushed. If this option is omitted, the current working directory is flushed.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm flush** command forces the Cache Manager to flush data cached from each file or directory specified with the **-path** option. All data cached from these files and directories is discarded. The next time the data is requested, the Cache Manager contacts the File Exporter to obtain the current version, along with new tokens and other associated status information.

This command does not discard any altered data in the cache not written to the central copy maintained by the File Exporter. It also does not affect data in the buffers of application programs.

It is also possible to flush all cached data that resides in the same fileset as a specific file or directory with the **cm flushfileset** command. To force the Cache Manager to update fileset-related information, use the **cm checkfilesets** command.

Privilege Required

No privileges are required.

EXAMPLES

The following command flushes the file **projectnotes**, which is in the current working directory, and all data from the subdirectory **plans** from the cache:

```
$ cm flush projectnotes plans/*
```

RELATED INFORMATION

Commands: **cm checkfilesets(8dfs)**, **cm flushfileset(8dfs)**.

cm flushfileset

Purpose

Forces the Cache Manager to discard data cached from filesets that contain specified files or directories

Synopsis

cm flushfileset[**-path** { *filename directory_name*}][**-help**]

OPTIONS

-path { *filename* **|** *directory_name***}**
Specifies a file or directory from each fileset to be flushed. A file for which a full pathname is not specified is assumed to reside in the current working directory. If this option is omitted, the fileset containing the current working directory is flushed.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm flushfileset** command forces the Cache Manager to flush data cached from filesets that contain each file or directory specified with the **-path** option. All data cached from these filesets is discarded. The next time the data is requested, the Cache Manager contacts the File Exporter to obtain the current version, along with new tokens and other associated status information.

This command does not discard any altered data in the cache not written to the central copy maintained by the File Exporter. It also does not affect data in the buffers of application programs.

It is also possible to flush data cached from specific files or directories with the **cm flush** command. To force the Cache Manager to update fileset-related information, use the **cm checkfilesets** command.

Privilege Required

No privileges are required.

EXAMPLES

The following command flushes data cached from the fileset containing the current working directory and the directory **reports**, both of which are at the same level in the file tree:

```
$ cm flushf.../reports
```

RELATED INFORMATION

Commands: **cm checkfilesets(8dfs)**, **cm flush(8dfs)**.

cm getcachesize

Purpose

Shows the current size of the cache, the amount of cache in use, and the type of cache

Synopsis

cm getcachesize[**-help**]

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **cm getcachesize** command displays the current size of the cache available to the Cache Manager and the amount in use when the command is issued. It also displays the type of cache in use on the machine. The command works both on machines using disk caching and on machines using memory caching.

The information displayed by the command comes from the kernel of the workstation on which the command is issued. On machines using disk caching, the current cache size may disagree with the default setting specified in the **CacheInfo** file if the cache size was set with the **cm setcachesize** command. Regardless of the type of caching (disk or memory) in use, the size may also disagree with the default setting if it was changed with the **dfsd** command.

Privilege Required

No privileges are required.

OUTPUT

The **cm getcachesize** command displays the following output:

```
DFS using amount of the cache's available size 1K byte (type) blocks.
```

In the output, *amount* is the number of kilobyte blocks the Cache Manager is currently using, **size** is the total number of kilobyte blocks available to the Cache Manager (the current cache size), and *type* is the type of cache (disk or memory) in use on the machine.

EXAMPLES

The following command shows the output on a machine with a 25,000 kilobyte disk cache:

```
$ cm getcachesize


DFS using 22876 of the cache's available 25000 1K byte (disk) blocks.
```

RELATED INFORMATION

Commands: **cm setcachesize(8dfs)**, **dfsd(8dfs)**.

Files: **CacheInfo(4dfs)**.

cm getdevok

Purpose

Shows whether device files from specified filesets are honored by the Cache Manager

Synopsis

cm getdevok[**-path** { *filename* | *directory_name*}**...**][**-help**]

OPTIONS

-path { *filename* **|** *directory_name***}**
Names a file or directory from each fileset whose device file status information is to be displayed. If this option is omitted, status information is displayed for the fileset containing the current working directory.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm getdevok** command reports whether the Cache Manager honors device files that reside in the indicated filesets. Indicate each fileset for which you want device file status information by specifying the name of a file or directory in the fileset with the **-path** option. This information comes from the kernel of the workstation on which the command is issued.

System administrators set whether device files are to be honored on a per-fileset and per-Cache-Manager basis with the **cm setdevok** command. By default, the Cache Manager does not honor device files from a fileset. (The UNIX kernel always honors device files stored in the **/dev** directory.)

Privilege Required

No privileges are required.

OUTPUT

The **cm getdevok** command first displays the line

Fileset *pathname* status:

In the output, *pathname* is the name of a file or directory specified with the **-path** option. For each specified file or directory, the following output values are possible for the fileset on which it resides:

device files allowed
Indicates that device files from the fileset are honored.

device files not allowed
Indicates that device files from the fileset are not honored.

cm: the fileset on which ′ *pathname***’ resides does not exist**
Indicates that the specified pathname is invalid.

EXAMPLES

The following command indicates that device files from the fileset that contains the directory */.../abc.com/fs/usr/jlw* are not honored by the Cache Manager:

```
$ cm getdevok /.../abc.com/fs/usr/jlw
/.../abc.com/fs/user/jlw status: device files not allowed
```

RELATED INFORMATION

Commands: **cm setdevok(8dfs)**.

cm getpreferences

Purpose

Displays the Cache Manager's preferences for File Server or Fileset Location (FL) Server machines

Synopsis

cm getpreferences[**-path** *filename*][**-numeric][-fldb][-help]**

OPTIONS

-path *filename*
: Specifies the full pathname of a file to which the command is to write the Cache Manager server preferences that it reports. If the specified file already exists, the command overwrites it. The command fails if the specified pathname names a directory. Omit this option to display the preferences on standard output (**stdout**).

-numeric
: Directs the command to display the IP addresses rather than the host names of the File Servers or FL Servers. Omit this option to display the host name (for example, **fs1.abc.com**) of each machine.

-fldb Directs the command to display the host names or IP addresses of the FL Servers and their respective ranks.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm getpreferences** command displays the current set of entries in a Cache Manager preference list. The Cache Manager preference list is created each time a Cache Manager is initialized with the **dfsd** command (each time the client machine is rebooted). Each Cache Manager maintains its own separate preference list. Each entry in the list consists of the IP address of an FL Server or File Server and an automatically assigned preference value. New entries are automatically added to the preference list as necessary when filesets are first referenced.

In operation, when the Cache Manager needs to contact an FL Server, it consults its list of FL Servers and attempts to contact a server at the address with the lowest-ranking value in the preference list. Similarly, when a Cache Manager needs to contact a File Server, it consults its preference list and contacts a suitable File Server at the address with the lowest-ranking value.

If the Cache Manager cannot access a server at the address with the lowest preference rank (because of a problem with either the machine or the network), the Cache Manager attempts to access a similar server at the address with the next lowest rank. It continues in this way until it either succeeds in accessing an appropriate server or determines that all such servers are unavailable.

By default, the Cache Manager assigns preferences that make sensible choices based on the location of servers. Therefore, you should adjust the default values only if there is a compelling reason. The default values force the Cache Manager to attempt to connect to servers in the following order:

1. The same machine as the client (default rank of 5000).
2. The same subnetwork as the client (default rank of 20000).
3. The same network as the client (default rank of 30000).
4. Different networks (default rank of 40000).

For example, a server on the same machine as the Cache Manager receives a rank of 5000, while a server on the same subnetwork receives a rank of 20000. The entry with the lowest-ranking value has the highest ″preference.″ Thus, a server with a preference value of 5000 will be chosen before a server with a rank of 20000.

Should two servers be assigned the same preference value, such as two File Servers on the same subnetwork both receiving a default value of 20000, the server with the lowest round-trip value is chosen. Each server is assigned a random round-trip value when the Cache Manager is initialized. The assigned round-trip value is always higher than the upper bound for stored actual round-trip values. This ensures that an actual round-trip value will always be chosen over assigned values. The **cm getpreferences** command does not display the round-trip value.

The **cm getpreferences** command displays information on standard output by default. Use the **-path** option to specify the complete pathname of a file to which the command is to write its output. If you include the -path option, the command displays no output on standard output.

Privilege Required

No privileges are required.

OUTPUT

The **cm getpreferences** command displays a separate line of output for each Cache Manager preference list entry. By default, each line consists of the host name of a File Server or FL Server followed by the preference value, as follows:

hostname rank

where *hostname* is the name of a File Server or FL Server, and *rank* is the rank associated with the machine. If the **-numeric** option is included with the command, the command displays the IP address, in dotted decimal format, instead of the machine's name. The command also displays the IP address of any machine whose name it cannot determine (for example, if a network outage prevents it from resolving the address into the name).

EXAMPLES

The following command displays the preference list entries associated with the Cache Manager on the local machine. The local machine belongs to the DCE cell named **dce.abc.com**; the ranks of the File Servers from the **dce.abc.com** cell are lower than the ranks of the File Servers from the foreign cell, **dce.def.com**. The command shows the IP addresses, not the names, of two machines from the foreign cell because it cannot currently determine their names.

```
$ cm getp
fs2.abc.com              20000
fs3.abc.com              30000
fs1.abc.com              20000
fs4.abc.com              30000
```

```
server1.def.com          40000
121.86.33.34             40000
server6.def.com          40000
121.86.33.37             40000
```

The following command displays the same Cache Manager's preference list entries, but the **-numeric** option is included with the command to display the IP addresses rather than the host names of all File Servers. The IP address of the local machine is **128.21.16.221**. The two File Servers on the same subnetwork as the local machine have a rank of 20000; the two File Servers on a different subnetwork in the same network as the local machine have a rank of 30000; the remaining File Servers are in a different network, so they have a rank of 40000. The round-trip value for each preference list entry (used to select a connection when multiple entries have the same rank) is not displayed by the command.

```
$ cm getp -n
128.21.16.214            20000
128.21.18.99             30000
128.21.16.212            20000
128.21.18.100            30000
121.86.33.41             40000
121.86.33.34             40000
121.86.33.36             40000
121.86.33.37             40000
```

RELATED INFORMATION

Commands: **cm setpreferences(8dfs)**.

cm getprotectlevels

Purpose

Returns the current DCE RPC authentication level settings for communications between the Cache Manager and File Servers

Synopsis

cm getprotectlevels[**-help**]

OPTIONS

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm getprotectlevels** command returns the current Cache Manager DCE RPC authentication level settings. The returned values include separate local and foreign cell settings for the initial and minimum authentication levels for communications with File Servers.

The Cache Manager and File Server default settings are such that communications occur at the Packet authentication level for the local cell.

The authentication bounds for the File Server itself are set through the **fxd** command. In addition to a general pair of upper and lower bounds for all communications between the File Server and Cache Manager, administrators can also set advisory bounds on a per fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level . Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset. You can display the current advisory RPC authentication bounds for a fileset through either the **fts lsfldb** or **fts lsft** commands.

Privilege Required

No privileges are required.

OUTPUT

The output consists of the following four lines:

```
Initial protection level in the local cell: level
Minimum protection level in the local cell: level
Initial protection level in non-local cells: level
Minimum protection level in non-local cells: level
```

Where *level* is one of the various DCE RPC authentication levels, whose possible values are

- **rpc_c_protect_level_default** - default : Use the DCE default authentication level.
- **rpc_c_protect_level_none** - none : Perform no authentication.
- **rpc_c_protect_level_connect** - connect : Authenticate only when the Cache Manager establishes a connection with the File Server.

- **rpc_c_protect_level_call** - call : Authenticate only at the beginning of each RPC received.
- **rpc_c_protect_level_pkt** - packet : Ensure that all data received is from the expected principal.
- **rpc_c_protect_level_pkt_integ** - packet integrity : Authenticate and verify that none of the of the data transferred has been modified.
- **rpc_c_protect_level_pkt_privacy** - packet privacy : Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

EXAMPLES

The following command returns the current authentication levels for communications between the Cache Manager and Files Servers:

```
$ cm getprotectlevels


Initial protection level in the local cell: rpc_c_protect_level_pkt
Minimum protection level in the local cell: rpc_c_protect_level_none
Initial protection level in non-local cells: rpc_c_protect_level_pkt_integ
Minimum protection level in non-local cells: rpc_c_protect_level_pkt
anonymous access to untrusted cells is permitted
```

RELATED INFORMATION

Commands: **cm setprotectlevels(8dfs)**, **fxd(8dfs)**, **dfsd(8dfs)**, **fts setprotectlevels(8dfs)**

cm getsetuid

Purpose

Shows the status of **setuid** programs from specified filesets

Synopsis

cm getsetuid[**-path**{ *filename* | *directory_name*}**...**][**-help**]

OPTIONS

-path { *filename* **|** *directory_name***}**
Names a file or directory from each fileset whose **setuid** permission is to be displayed. If this option is omitted, permission information is displayed for the fileset containing the current working directory.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm getsetuid** command reports whether the Cache Manager allows **setuid** programs from the indicated filesets to run with **setuid** permission. Indicate each fileset whose **setuid** permission is desired by specifying the name of a file or directory in the fileset with the **-path** option. This information comes from the kernel of the workstation on which the command is issued.

Note that **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

Because **setgid** programs on filesets are enabled or disabled along with **setuid** programs, this command also reports the status of **setgid** programs on the indicated filesets. System administrators set **setuid** and **setgid** status on a per-fileset and per-Cache Manager basis with the **cm setsetuid** command. By default, the Cache Manager does not allow **setuid** programs from a fileset to execute with **setuid** permission.

Privilege Required

No privileges are required.

OUTPUT

The **cm getsetuid** command first displays the line

```
Fileset pathname status:
```

In the output, *pathname* is the name of a file or directory specified with the **-path** option. For each specified file or directory, the following output values are possible for the fileset on which it resides:

setuid allowed
Indicates that **setuid** and **setgid** programs from the fileset are enabled.

no setuid allowed
Indicates that **setuid** and **setgid** programs from the fileset are disabled.

cm: the fileset on which ′ *pathname*’ resides does not exist
Indicates that the specified pathname is invalid.

EXAMPLES

The following command indicates that **setuid** and **setgid** programs from the fileset that contains the directory */.../abc.com/fs/usr/jlw* are disabled:

```
$ cm getsetuid /.../abc.com/fs/usr/jlw
Fileset /.../abc.com/fs/usr/jlw status: no setuid allowed
```

RELATED INFORMATION

Commands: **cm setsetuid(8dfs)**.

cm help

Purpose

Shows syntax of specified **cm** commands or lists functional descriptions of all **cm** commands

Synopsis

cm help[**-topic** *string*][**-help**]

OPTIONS

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **flush**, not **cm flush**). If this option is omitted, the output provides a short description of all **cm** commands.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm help** command displays the first line (name and short description) of the online help entry for every **cm** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **cm apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

OUTPUT

The online help entry for each **cm** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with **Usage:**, lists the command options in the prescribed order.

EXAMPLES

The following command displays the online help entry for the **cm flush** command:

```
$ cm help flush
```

```
cm flush: flush file from cache
Usage: cm flush [-path {<filename> | <directory_name>}...] [-help]
```

RELATED INFORMATION

Commands: **cm apropos(8dfs)**.

cm lscellinfo

Purpose

Shows database server machines in cells known to the Cache Manager

Synopsis

cm lscellinfo[**-help**]

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **cm lscellinfo** command formats and displays the Cache Manager's kernel-resident list of Fileset Location Database (FLDB) machines in its home cell and any foreign cells the Cache Manager has accessed. This information comes from the kernel of the workstation on which the command is issued.

Privilege Required

No privileges are required.

OUTPUT

The output contains one line for the local cell and one line for each cell listed in the kernel that the Cache Manager has accessed. Each cell name is followed by a list of its database server machines (referred to as *hosts*). In a multihomed server environment (an FLDB machine can have up to four IP addresses listed in the Cache Manager's preferences), *hosts* corresponds to the IP addresses or host names that the Cache Manager is currently using to access each particular FLDB machine. Therefore, the command output lists only one machine name for each FLDB machine.

EXAMPLES

The following command shows output for several cells:

```
$ cm lscellinfo
Cell abc.com on hosts fs2.abc.com
Cell state.edu on hosts fs11.fs.state.edu
```

cm lsstores

Purpose

Lists filesets that contain data the Cache Manager cannot write back to a File Server machine

Synopsis

cm lsstores [**-help**]

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **cm lsstores** command lists the fileset ID numbers of filesets that contain data the Cache Manager cannot write back to a File Server machine. This information comes from the kernel of the workstation on which the command is issued.

On occasion, a File Server machine may be unavailable to the Cache Manager (possibly because the File Server machine is down or because a network problem prevents the Cache Manager from contacting the machine). In such cases, the Cache Manager cannot write data back to the File Server machine. The Cache Manager displays a message on the screen to notify the user that it cannot write the data to the unavailable machine. If possible, it also returns a failure code to the application program using the data.

The Cache Manager keeps the unstored data in the cache and continues to attempt to contact the File Server machine until it can store the data. (The frequency with which it attempts to reach a File Server machine is defined with the **-pollinterval** option of the **fxd** command issued on that File Server machine.) In the meantime, corrective measures can be taken to alleviate the problem that prevents the data from being stored; for example, the File Server machine can be restarted. Once the problem is alleviated, the Cache Manager can reach the File Server machine and store the data.

The Cache Manager discards unstored data only when

- It needs to make room in the cache for other data. Given an average-sized cache with average usage, the Cache Manager rarely needs to discard unstored data.
- The **cm resetstores** command is issued to force it to discard unstored data from the cache.

Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

Privilege Required

No privileges are required.

OUTPUT

If the Cache Manager cannot store data to one or more filesets, the command displays the fileset ID number of each fileset to which data cannot be stored. If the Cache Manager has been able to store all data, the command displays the following message:

```
No failed stores are being retried.
```

RELATED INFORMATION

Commands: **cm resetstores(8dfs)**, **fxd(8dfs)**.

cm resetstores

Purpose

Cancels attempts by the Cache Manager to contact unavailable File Server machines and discards all data the Cache Manager cannot store to such machines

Synopsis

cm resetstores[**-help**]

OPTIONS

-help Prints the online help for this command.

DESCRIPTION

The **cm resetstores** command cancels the Cache Manager's continued attempts to contact unavailable File Server machines. *All* data that the Cache Manager cannot store to such File Server machines is discarded; there is no way to selectively discard individual files or data from specific filesets.

Occasionally, a File Server machine may be unavailable to the Cache Manager (possibly because the File Server machine is down or because a network problem prevents the Cache Manager from contacting the machine). In such cases, the Cache Manager cannot write data back to the File Server machine. The Cache Manager displays a message on the screen to notify the user that it cannot write the data to the unavailable machine. If possible, it also returns a failure code to the application program using the data.

The Cache Manager keeps the unstored data in the cache and continues to attempt to contact the File Server machine until it can store the data. (The frequency with which it attempts to reach a File Server machine is defined with the **-pollinterval** option of the **fxd** command issued on that File Server machine.) In the meantime, corrective measures can be taken to alleviate the problem that prevents the data from being stored; for example, the File Server machine can be restarted. Once the problem is alleviated, the Cache Manager can reach the File Server machine and store the data.

The Cache Manager discards unstored data only when

- It needs to make room in the cache for other data. Given an average-sized cache with average usage, the Cache Manager rarely needs to discard unstored data.
- The **cm resetstores** command is issued to force it to discard unstored data from the cache.

Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

Note that the **cm resetstores** command affects only data that could not be stored to a File Server machine; it does not affect other data in the cache. Nonetheless, be cautious when issuing the **cm resetstores** command. Issue the **cm lsstores** command first to list the fileset ID numbers of filesets that contain data the Cache Manager cannot write to a File Server machine; examine the output of the

command to be sure that you know from which filesets unstored data will be discarded. (You may also be able to use this information to ensure that unstored data from the indicated filesets can safely be discarded.)

Privilege Required

The issuer must be logged in as **root** on the local machine.

RELATED INFORMATION

Commands: **cm lsstores(8dfs)**, **fxd(8dfs)**.

cm setcachesize

Purpose

Sets the size of a disk cache

Synopsis

cm setcachesize {**-size** *kbytes* | **-reset** } [**-help**]

OPTIONS

-size *kbytes*
Specifies the number of 1-kilobyte blocks the Cache Manager can use for the cache. The smallest allowable value is 1. Specifying a value of 0 (zero) sets the cache size to the default specified in the **CacheInfo** file. Use this option or use the **-reset** option.

-reset Returns the cache size to the value set when the machine was last booted. Use this option or use the **-size** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm setcachesize** command changes the amount of local disk space the Cache Manager uses for its data cache. Specify a number of kilobyte blocks. Do not set the cache size to exceed 85 of the actual disk space available for the cache directory; the cache implementation itself requires a small amount of room on the partition. *Do not use this command on a machine using a memory cache.*

The cache size cannot be set to a value less than twice the value of the chunk size in use by the Cache Manager. If a value smaller than twice the chunk size is specified with the **-size** option, the following message is displayed:

```
path: Cache size of size is too small; value was rounded up.
```

In the message, *path* is the specified path to the **cm** program (usually just **cm**) and **size** is the size, in kilobytes, specified with the command. The standard message reporting the new cache size (the size to which the cache was rounded) is then displayed; see the section on output for an example of the message.

To return the cache size to the default value specified in the **CacheInfo** file, specify 0 (zero) as the number of kilobyte blocks with the **-size** option. To return the cache size to the value set when the machine was last booted, use the **-reset** option instead of the **-size** option; the **-reset** option also sets the size to the amount specified in the **CacheInfo** file unless the **-blocks** option was used with the **dfsd** command to override the **CacheInfo** value, in which case the value set with the **dfsd** command is used.

The **cm getcachesize** command displays the current cache size and the amount of space in use for both disk and memory caches. It also reports the type of cache (disk or memory) in use.

Privilege Required

The issuer must be logged in as **root** on the local machine.

OUTPUT

The following message is displayed whenever this command is used to set the cache size:

path: New cache size set: size.

In the message, *path* is the specified path to the **cm** program (usually just **cm**) and **size** is the new cache size, in kilobytes.

EXAMPLES

The following command sets the cache size to 25,000 kilobyte blocks:

```
# cm setca 25000
cm: New cache size set: 25000.
```

The following command resets the cache size to the value set when the machine was last booted (50,000 kilobyte blocks, in this case):

```
# cm setca -r
cm: New cache size set: 50000.
```

RELATED INFORMATION

Commands: **cm getcachesize(8dfs)**, **dfsd(8dfs)**.

Files: **CacheInfo(4dfs)**.

cm setdevok

Purpose

Specifies whether device files from specified filesets are honored by the Cache Manager

Synopsis

cm setdevok[**-path**{ *filename directory_name*}][**-state** {on | off}][**-help**]

OPTIONS

-path { *filename* | *directory_name*}
Names a file or directory from each fileset whose device file status is to be changed. If this option is omitted, the status is changed for the fileset containing the current working directory.

-state Specifies whether device files from the filesets indicated with **-path** are to be honored. Specify **on** with this option to honor device files from the indicated filesets; specify **off** with this option to prevent device files from the indicated filesets from being honored. If this option is omitted, device files from the filesets are honored. (The command has no effect if device files were already honored.)

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm setdevok** command specifies whether device files from the indicated filesets are honored by the Cache Manager. Indicate each fileset whose device files are to be honored or not honored by specifying the name of a file or directory in the fileset with the **-path** option. Device files are honored on a per-fileset and per-Cache Manager basis. This command is commonly included in a start-up file (**/etc/rc** or its equivalent) to honor device files at machine startup.

If **on** is specified with the **-state** option, or if the **-state** option is omitted, the Cache Manager honors device files from the indicated filesets. If **off** is specified with the **-state** option, the Cache Manager does not honor device files from the indicated filesets. By default, the Cache Manager does not honor device files from a fileset. (The UNIX kernel always honors device files stored in the **/dev** directory.)

The **cm getdevok** command displays whether the Cache Manager honors device files from indicated filesets.

Privilege Required

The issuer must be logged in as **root** on the local machine.

EXAMPLES

The following command causes device files that reside on the fileset that contains the directory */.../abc.com/fs/usr/jlw* to be honored:

```
# cm setdevok /.../abc.com/fs/usr/jlw
```

RELATED INFORMATION

Commands: **cm getdevok(8dfs)**.

cm setpreferences

Purpose

Sets the Cache Manager's preferences for the File Server or File Location (FL) Server machines

Synopsis

cm setpreferences[**-server** *machine rank*][**-path** *filename*][**-stdin**][**-fldb**] [**-help**]

OPTIONS

-server *machine rank*
: Specifies File Server or FL Server preference entries, with each entry consisting of a machine specification (a host name or IP address) and a preference rank. Separate each machine specification and each rank with one or more spaces. By default, the **-server** option specifies File Server machine entries; add the **-fldb** option to specify FL Server machine entries. Each server machine can have multiple preference entries, with each entry having a unique host name or IP address. Refer to the ″Specifying Preferences″ section of this reference page for information about specifying File Server or FL Server entries.

-path *filename*
: Specifies the full pathname of a file from which the command is to read preference entries. Each entry consists of a File Server or FL Server machine specification (a host name or IP address) and its respective rank. Separate each machine specification from its rank with one or more spaces, and include each paired machine specification and rank on a separate line. Refer to the ″Specifying Preferences″ section of this reference page for information about specifying File Server or FL Server entries.

-stdin Directs the command to read File Server or FL Server preference entries from standard input (**stdin**). Each entry must consist of a machine specification (either a host name or IP address) and a ranking value. Separate each machine specification and each rank with one or more spaces. Refer to the ″Specifying Preferences″ section of this reference page for information about specifying File Server or FL Server entries.

-fldb Directs the command to consider the servers specified in the **-server** option as FL Servers.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm setpreferences** command can be used to add preference entries to a Cache Manager preference list or modify ranking values for existing preference entries. The Cache Manager preference list is created each time a Cache Manager is initialized with the **dfsd** command (each time the client machine is rebooted). Each Cache Manager maintains its own separate preference list. Each entry in the list consists of the IP address of an FL Server or File Server and an automatically assigned preference value. New entries are automatically added to the preference list as necessary when filesets are first referenced.

In operation, when the Cache Manager needs to contact an FL Server, it consults its list of FL Servers and attempts to contact a server at the address with the lowest-ranking value in the preference list. Similarly, when a Cache Manager needs to contact a File Server, it consults its preference list and contacts a suitable File Server at the address with the lowest-ranking value.

If the Cache Manager cannot access a server at the address with the lowest preference rank (because of a problem with either the machine or the network), the Cache Manager attempts to access a similar server at the address with the next-lowest rank. It continues in this way until it either succeeds in accessing an appropriate server or determines that all such servers are unavailable.

By default, the Cache Manager assigns preferences that make sensible choices based on the location of servers. Therefore, you should adjust the default values only if there is a compelling reason to do so. The default values force the Cache Manager to attempt to connect to servers in the following order:

- The same machine as the client (default rank of 5000).
- The same subnetwork as the client (default rank of 20000).
- The same network as the client (default rank of 30000).
- Different networks (default rank of 40000).

For example, a server on the same machine as the Cache Manager receives a rank of 5000, while a server on the same subnetwork receives a rank of 20000. The entry with the lowest-ranking value has the highest ″preference.″ Thus, a server with a preference value of 5000 will be chosen before a server with a rank of 20000.

Should two servers be assigned the same preference value, such as two File Servers on the same subnetwork both receiving a default value of 20000, the server with the lowest round-trip value is chosen. Each server is assigned a random round-trip value when the Cache Manager is initialized. The assigned round-trip value is always higher than the upper bound for stored actual round-trip values. This ensures that an actual round-trip value will always be chosen over assigned values.

The Cache Manager stores its preferences in the kernel of the local machine. The preferences are lost each time the Cache Manager is initialized. You can include the **cm setpreferences** command in a machine's initialization file to load a predefined collection of server preferences when the machine is rebooted.

Specifying Preferences

Using the **cm setpreferences** command, you specify Cache Manager preference entries as pairs of values. The first value of the pair is the machine specification (either the host name or IP address in dotted decimal format) of a File Server or FL Server; the second value is the preference rank (an integer in the range from 1 to 65,534). The FLDB can contain up to four addresses for each server machine (although the machine can have more connections); therefore, the Cache Manager preference list will normally have up to four entries for a given server machine.

You can specify the following preference entries:

- On the command line via the **-server** option. Use this option to tune the preferences manually in response to system or network adjustments.

- From a file via the **-path** option. Use this option to configure one or more Cache Managers with a fixed set of preferences. You can use the **cm getpreferences** command to generate a file of preferences that has the proper format.
- From standard input via the **-stdin** option. Use this option to pipe preferences to the command from a user-defined process that generates preferences in an acceptable format.

The **-server**, **-path**, and **-stdin** options are not mutually exclusive. You can include any combination of these options with the command to provide input from multiple sources. Note that the command does not verify host names or IP addresses specified with any of its options. You can add a preference for an invalid host name or IP address; the Cache Manager stores invalid preferences in the kernel, but it ignores them (the Cache Manager never needs to consult such preferences).

Privilege Required

The issuer must be logged in as **root** on the local machine.

OUTPUT

By default, the **cm setpreferences** command displays no output.

EXAMPLES

The following command uses the **-server** option to set the Cache Manager's preferences for the File Servers named **fs3.abc.com** and **fs4.abc.com**, the latter of which is specified by IP address. The two File Servers reside in a different subnetwork that is in the same network as the local machine. Therefore, the Cache Manager assigned each a default rank of 30,000. To make the Cache Manager prefer these File Servers over File Servers in other subnetworks, the **cm setpreferences** command is used to assign these machines ranks of 25,000.

```
# cm setp -se fs3.abc.com 25000 128.21.18.100 25000
```

The following command uses the **-server** option to set the Cache Manager's preferences for the same two File Servers, but it also uses the **-path** option to read a collection of preferences from a file that resides on the local machine at **/etc/cm.prefs**:

```
# cm setp -se fs3.abc.com 25000 128.21.18.100 25000 -p
/etc/cm.prefs
```

The file **/etc/cm.prefs** has the following contents and format:

```
128.21.16.214 7500
128.21.16.212 7500
121.86.33.41 39000
121.86.33.34 39000
121.86.33.36 41000
121.86.33.37 41000
```

The following command uses the **-stdin** option to read preferences from standard input. The preferences are piped to the command from a program, **calc_prefs**, which was written by the issuer to calculate preferences based on values significant to the local cell.

```
# calc_prefs | cm setp -stdin
```

RELATED INFORMATION

Commands: **cm getpreferences(8dfs)**, **dfsd(8dfs)**.

cm setprotectlevels

Purpose

Adjusts DCE remote procedure call (RPC) authentication levels for communications between the Cache Manager and File Servers

Synopsis

cm setprotectlevels[**-initiallocalprotectlevel** *level*] [**-minlocalprotectlevel** *level*] [**-initialremoteprotectlevel** *level*] [**-minremoteprotectlevel** *level*] [**-anonaccess** {on | off}[-help]

OPTIONS

-initiallocalprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-minlocalprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-initialremoteprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-minremoteprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

-anonaccess
: Allows access to untrusted cells.

DESCRIPTION

The **cm setprotectlevels** command adjusts the DCE RPC security level for RPCs sent between a Cache Manager and DFS File Servers. The command adjusts two levels: an initial DCE RPC security level used as a starting point in security level negotiations between the Cache Manager and a File Server and the minimum DCE RPC security level the Cache Manager will accept for such communications. Two

sets of these levels are maintained: one set specifies the security levels for communications with File Servers within the local cell and the other set specifies the security levels for communications with File Servers within foreign cells. Both sets of security levels are initially set through the **dfsd** command.

In operation, the Cache Manager and File Server interact to arrive at a mutually acceptable authentication level for communications. The negotiation starts with an RPC using the initial authentication level sent from the Cache Manager to the File Server. If the initial authentication level is outside the minimum or maximum bounds set at the File Server, the File Server returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Server requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Server). Once the Cache Manager and File Server have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Server.

The authentication bounds for communications at the File Server itself is set through the **fxd** command. The Cache Manager and **fxd** default settings are such that communications occur at the Packet authentication levelfor the local cell.

In addition to a general pair of upper and lower bounds for all communications between the File Server and Cache Manager, administrators can also set advisory bounds on a per fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level (they may be enforced in a future version of DFS). Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset.

Note that the use of this command does not preclude communications with File Servers running earlier versions of DFS.

The various authentication levels are set by specifying either an integer value between 0 and 6, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **0** or **default** or **rpc_protect_level_default**: Use the DCE default authentication level.
- **1** or **none** or **rpc_protect_level_none**: Perform no authentication.
- **2** or **connect** or **rpc_protect_level_connect**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **3** or **call** or **rpc_protect_level_call**: Authenticate only at the beginning of each RPC received.
- **4** or **pkt** or **rpc_protect_level_pkt**: Ensure that all data received is from the expected host.
- **5** or **pkt_integrity** or **rpc_protect_level_pkt_integrity**: Authenticate and verify that none of the data transferred has been modified.
- **6** or **pkt_privacy orrpc_protect_level_pkt_privacy**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

Privilege Required

The issuer must be logged in as **root** on the local machine.

EXAMPLES

The following command sets the following authentication values:

1. The initial authentication level for communications with File Servers in the local cell is set to packet integrity.
2. The minimum authentication level for communications with File Servers in the local cell is set to packet.
3. The initial authentication level for communications with File Servers in foreign cells is set to packet privacy.
4. The minimum authentication level for communications with File Servers in foreign cells is set to packet privacy.

```
$ cm setprotectlevels -initiallocalprotectlevel 5 -minlocalprotectlevel 4
-initialremoteprotectlevel 6 -minremoteprotectlevel 6
```

RELATED INFORMATION

Commands: **cm getprotectlevels(8dfs)**, **fxd(8dfs)**, **dfsd(8dfs)**, **fts setprotectlevels(8dfs)**

cm setsetuid

Purpose

Enables or disables **setuid** programs from specified filesets

Synopsis

cm setsetuid[**-path**{ *filename directory_name*}][**-state** {on | off}][**-help**]

OPTIONS

-path { *filename* **|** *directory_name***}**
Names a file or directory from each fileset whose **setuid** status is to be changed. If this option is omitted, the status is changed for the fileset containing the current working directory.

-state Allows or disallows **setuid** programs from the filesets indicated with **-path** to execute with **setuid** permission. Specify **on** with this option to allow **setuid** programs from the indicated filesets to execute with **setuid** permission; specify **off** with this option to disallow **setuid** programs from the indicated filesets to execute with **setuid** permission. If this option is omitted, **setuid** programs from the filesets are allowed to execute with **setuid** permission. (The command has no effect if **setuid** permission was already enabled.)

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm setsetuid** command enables **setuid** programs from the indicated filesets to execute with **setuid** permission or prevents them from executing with **setuid** permission. Indicate each fileset whose **setuid** permission is to be enabled or disabled by specifying the name of a file or directory in the fileset with the **-path** option. The permissions are enabled or disabled on a per-fileset and per-Cache Manager basis. This command is commonly included in a start-up file (**/etc/rc** or its equivalent) to enable **setuid** programs at machine startup.

If **on** is specified with the **-state** option, or if the **-state** option is omitted, the Cache Manager allows **setuid** programs from the indicated filesets to execute with **setuid** permission. If **off** is specified with the **-state** option, the Cache Manager does not allow **setuid** programs from the indicated filesets to execute with **setuid** permission. By default, the Cache Manager does not allow **setuid** programs from a fileset to execute with **setuid** permission.

A **setuid** program is indicated by setting a mode bit associated with an executable file. While a **setuid** program executes, the person executing the program is treated as if he or she is the owner of the program. The effective user identification number (UID) of the executing program is the UID of the person who owns the program, not the UID of the person who initiated the program's execution. Thus, the person executing the program is granted the same permissions as the person who owns the program for the duration of the program's execution.

Note that **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot

change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

The **cm setsetuid** command enables or disables **setgid** programs from the indicated filesets at the same time that it changes the status of **setuid** programs. The **cm getsetuid** command displays whether the Cache Manager allows **setuid** and **setgid** programs from indicated filesets to execute.

Privilege Required

The issuer must be logged in as **root** on the local machine.

EXAMPLES

The following command enables **setuid** and **setgid** programs that reside on the fileset containing the directory */.../abc.com/fs/usr/jlw*:

```
# cm setsetuid /.../abc.com/fs/usr/jlw
```

RELATED INFORMATION

Commands: **cm getsetuid(8dfs)**.

cm statservers

Purpose

Checks the statuses of File Server machines

Synopsis

cm statservers{**-cell** *cellname***-all** }[**-fast**][**-help**]

OPTIONS

-cell *cellname*
Specifies the name of the specific cell the Cache Manager is to probe for the status of each File Server machine it has contacted or has attempted to contact from that cell. The Cache Manager probes only machines in the specified cell. Use this option or use the **-all** option; omit both options to direct the Cache Manager to probe only machines in the local cell.

-all Directs the Cache Manager to probe all of the machines it has contacted in all cells. Use this option or use the **-cell** option; omit both options to direct the Cache Manager to probe only machines in the local cell.

-fast Directs the Cache Manager to display its current list of contacted File Server machines without probing the machines. This option can be combined with the **-cell** or **-all** option; it can also be used if both the **-cell** and -all options are omitted.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm statservers** command lists all File Server machines in the indicated cells that meet the following two conditions:

- The Cache Manager has been in contact with the File Exporter running on the machine and needs to contact it in the future (probably because the Cache Manager is holding tokens for data on that File Server machine).
- The File Exporter on the machine is not currently responding to the Cache Manager's probes (implying that it is not responding to the Cache Manager's requests for data either).

The Cache Manager maintains a list of File Server machines that meet the first condition, updating the list periodically by attempting to contact the File Exporter on each machine in the list. When a machine does not respond to a probe, the Cache Manager marks it as nonfunctioning. If a machine that previously did not respond begins to respond again, the Cache Manager erases the mark. The Cache Manager maintains this information in the kernel of the local machine.

Without the **-fast** option, this command forces the Cache Manager to update its information immediately (rather than waiting the standard interval). The Cache Manager probes the File Exporters on the machines in the specified cells, records those that do not respond, and reports the results. If you include the **-fast** option, the Cache Manager displays the list of nonfunctioning machines that it has at the time the command is issued; it does not probe the machines again.

By default, the Cache Manager probes machines in the local cell only. If the **-all** option is used, the Cache Manager probes all machines (from all cells) that meet the first condition. If a *cellname* is specified with the **-cell** option, the Cache Manager probes only the machines in that cell.

The execution of this command can be lengthy if a number of machines in the Cache Manager's list are unresponsive when the command is issued. The Cache Manager waits a standard timeout period before concluding that a File Exporter is not responding; this allows for the possibility of slow cross-network communication. If it is important that the command shell prompt return quickly, run this command in the background. It is harmless to interrupt the command (with **<Ctrl-c>** or another interrupt signal).

This command does not check the statuses of all File Server machines in a cell. The Cache Manager probes only those machines that meet the first condition in the previous list.

Privilege Required

No privileges are required.

OUTPUT

If the Cache Manager gets a response from all of the machines that it probes (that is, all such machines are functioning normally), the command displays the following output:

```
All servers are running.
```

This message does not imply that all File Server machines in the specified cells are running; it implies only that those machines that the Cache Manager probed are running.

If one or more machines fail to respond to the Cache Manager's probes within the timeout period, the command displays the following output:

```
These servers are still down: hostname
```

where *hostname* is the name of each File Server machine that fails to respond.

In a multihomed server environment (a File Server machine can have four IP addresses listed in the Cache Manager's preferences), the *hostname* corresponds to the host name or IP address that the Cache Manager is currently using to access each File Server machine. The output does not contain multiple machine names for the same File Server machine.

EXAMPLES

The following command uses the **-fast** option to view the Cache Manager's current list of unresponsive machines belonging to the local cell rather than waiting for the Cache Manager to probe them again. The output indicates that all machines responded to the most recent probes.

```
$ cm statservers -f
```

```
All servers are running.
```

The following command checks all File Server machines from which the Cache Manager has cached data, regardless of the cell in which a machine resides. The command reports that the machines named **fs1.abc.com** and **fs3.state.edu** did not respond to the Cache Manager's probes. The **&** (ampersand) is used to execute the command in the background.

```
$ cm statservers -all &


These servers are still down: fs1.abc.com fs3.state.edu
```

RELATED INFORMATION

Commands: **cm lsstores(8dfs)**, **cm whereis(8dfs)**.

cm sysname

Purpose

Reports or sets the CPU/OS type

Synopsis

cm sysname[**-newsys** *sysname*][**-help**]

OPTIONS

-newsys *sysname*
Specifies the new setting of the CPU/Operating System (@**sys**) variable for the machine on which it is issued. If this option is omitted, the output shows the current setting of the variable.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm sysname** command displays the current setting of the @**sys** variable or sets the variable on a client machine. If the **-newsys** option is omitted, the command reports the current setting of the @**sys** variable. If the **-newsys** option is included, the command sets the variable to the specified CPU/OS type. The value of the variable is displayed from or set in the kernel of the client machine on which the command is issued.

The Cache Manager's main use of the @**sys** variable is in pathnames used in symbolic links. As the Cache Manager interprets pathnames, it substitutes the value of the indicator for any occurrence of @**sys**. (Use the @**sys** variable sparingly; it can make the effect of changing directories confusing.)

Privilege Required

To view the current setting of @**sys** (without the **-newsys** option), no privileges are required. To change the setting of @**sys** (with the **-newsys** option), you must be logged in as **root** on the local machine.

OUTPUT

If the **-newsys** option is not specified, the output reports the system type in the following format:

```
Current sysname is 'system_type'.
```

EXAMPLES

The following command shows the output produced on a machine running OSF/1:

```
$ cm sys
Current sysname is 'pmax_osf1'.
```

The following commands set the system type on a machine running AIX 3.2 and use it in a symbolic link from the **/usr/local** directory on the local machine to a directory in the DFS filespace:

```
# cm sys -new rs_aix41
# ln -s /.../abc.com/fs/@sys/usr/local /usr/local
# ls -l /usr/local

lrwxrwxrwx 1 root 34 May 31 1993 /usr/local ->
/.../abc.com/fs/@sys/usr/local

# cd /usr/local
# pwd

/.../abc.com/fs/rs_aix41/usr/local
```

cm whereis

Purpose

Reports names of File Server machines that house specified files or directories

Synopsis

cm whereis [**-path** { *filename* | *directory_name*}**...**][**-help**]

OPTIONS

-path *filename* **or** *directory_name*
: Specifies the pathname of each file or directory whose location is to be reported. Each file or directory must reside in DFS, not on a local disk. If a full pathname is not provided, the file or directory is assumed to reside in the current working directory. If this option is omitted, the current working directory is used.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **cm whereis** command displays information about the location of each file or directory indicated with the **-path** option. The command reports the name of the cell in which the file or directory exists, the name of the fileset in which it resides, and the name of each File Server machine that houses a copy of the fileset. This information comes from the kernel of the workstation on which the command is issued.

Privilege Required

No privileges are required.

OUTPUT

The output includes a separate line displaying the following information about each file or directory specified with the **-path** option:

```
File 'filename' resides in the cell 'cellname' in fileset'
fileset_name' on host(s) 'hostname'.
```

where:

filename
: Specifies the complete pathname of a file or directory specified with the **-path** option.

cellname
: Specifies the name of the cell in which the file or directory is located.

fileset_name
: Specifies the name of the fileset in which the file or directory is located.

hostname
: Specifies the name of the File Server machine on which the fileset is located. If the fileset is a read/write or backup fileset, only one machine name is displayed; if the fileset is a read-only fileset, multiple machine

names can be displayed. However, only one machine name is displayed for each File Server machine. (The Cache Manager can have up to four preferences for each File Server machine, with each preference having a different host name or IP address.)

EXAMPLES

The following command indicates that the directory named */.../abc.com/fs/bin/sysfile* is located in a replicated fileset on the File Server machines named **fs1**, **fs3**, and **fs6**, all of which are located in the cell named *abc.com*:

```
$ cm whereis /.../abc.com/fs/bin/sysfile


File '/.../abc.com/fs/bin/sysfile' resides in the cell 'abc.com',
in fileset 'sysfile.bin', on hosts fs1.abc.com, fs3.abc.com,
fs6.abc.com.
```

RELATED INFORMATION

Commands: **cm statservers(8dfs)**.

dfsbind

Purpose

Provides user-space information to the Cache Manager and File Exporter

Synopsis

dfsbind[**-expressprocs** *number_of_express_daemons*][**-regularprocs** *number_of_regular_daemons*][**-junctionlife** *seconds_to_live*][**-prefixlife** *seconds_to_live*][**-notfoundlife** *seconds_to_live*][**-debug**][**-help**][**-foreground**]

OPTIONS

-expressprocs *number_of_express_daemons*
Specifies the number of express processes (user-space threads) allocated to handling requests for security information that do not require a substantial amount of time. By default, **dfsbind** uses one express process. Use this option to increase the number of express processes if the local machine encounters a large number of timeout errors. Specify an integer greater than 0 (zero) to indicate the number of express processes.

-regularprocs *number_of_regular_daemons*
Specifies the number of regular processes (user-space threads) allocated to handling requests for CDS pathname resolution and requests for security information that may require significant time. By default, **dfsbind** uses one regular process. Use this option to increase the number of regular processes if the local machine experiences a large number of timeout errors. Specify an integer greater than 0 (zero) to indicate the number of regular processes.

-junctionlife *seconds_to_live*
Specifies the length of time for which information cached about Fileset Database machines for a cell remains valid. When **dfsbind** retrieves this information from the DFS junction of a cell, it sends the information, along with a *time to live* (TTL), to the Cache Manager. The TTL specifies the length of time for which the Cache Manager is to consider the information valid. The Cache Manager caches the information and the TTL. It continues to recognize the information as valid until the TTL expires, after which it asks **dfsbind** to refresh the information the next time it needs it.

By default, **dfsbind** assigns a TTL of 24 hours to information about Fileset Database machines. This option can be used to change the TTL that **dfsbind** assigns to such information. Specify an integer greater than or equal to 30 to indicate the new TTL in seconds.

Note: *This option has an effect only on DFS client machines, where it is useful primarily for debugging purposes.*

-prefixlife *seconds_to_live*
Specifies the length of time for which information cached about a pathname that is a valid DFS junction name prefix remains valid. When **dfsbind** successfully traverses a given path but the path is not a DFS junction name, it sends the Cache Manager the valid pathname along with a TTL. The Cache Manager caches the information and the TTL, continuing to recognize the information as valid until the TTL expires; it then contacts **dfsbind** to refresh the information the next time it needs it.

By default, **dfsbind** assigns a TTL of 24 hours to information about pathnames that are valid DFS junction name prefixes. This option can be used to change the TTL that **dfsbind** assigns to such information. Specify an integer greater than or equal to 30 to indicate the new TTL in seconds.

Note: *This option has an effect only on DFS client machines, where it is useful primarily for debugging purposes.*

-notfoundlife *seconds_to_live*
Specifies the length of time for which information cached about an invalid pathname remains valid. When **dfsbind** cannot traverse a given path, it sends the Cache Manager the invalid pathname along with a TTL. The Cache Manager caches the information and the TTL, considering the information valid until the TTL expires; it then contacts **dfsbind** to refresh the information the next time it needs it.

By default, **dfsbind** assigns a TTL of 1 hour to information about invalid pathnames. This option can be used to change the TTL that **dfsbind** assigns to such information. Specify an integer greater than or equal to 30 to indicate the new TTL in seconds.

Note: *This option has an effect only on DFS client machines, where it is useful primarily for debugging purposes.*

-debug
Provides debugging information about the execution of the command. The primary usage of the information is to ensure that the process is executing properly. If this option is specified, the process does not automatically place itself in the background once it starts.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with **dfsbind**. See the **bos_help(8dfs)** and **bos_apropos(8dfs)** reference pages for examples using these commands.

-foreground
Causes **dfsbind** to run in the foreground instead of automatically placing itself in the background when it starts. This option can be used to place **dfsbind** under control of the **bosserver** process.

DESCRIPTION

The **dfsbind** command starts the **dfsbind** process, which provides user-space services to the Cache Manager on a DFS client machine or the File Exporter on a DFS File Server machine. (The Cache Manager and the File Exporter reside in the kernels of their respective machines.) The binary file for the **dfsbind** command resides in *dcelocal***/bin/dfsbind**. By default, the process automatically places itself in the background after it starts.

The **dfsbind** process must be run on all client machines and File Server machines. A machine that runs the Cache Manager (which is initialized by the **dfsd** command) and the **dfsbind** process is considered a DFS client machine. A machine that runs the Fileset Server (**ftserver** process), the File Exporter (which is initialized by the **fxd** command), and the **dfsbind** process is considered a DFS File Server machine.

On either type of machine, the **dfsbind** command is usually added to the proper start-up file (**/etc/rc** or its equivalent) rather than entered at the command shell

prompt. On a client machine, the **dfsbind** process must be run before the **dfsd** process in a start-up file; on a File Server machine, it must be run before the **fxd** process in a start-up file.

On a client machine, the **dfsbind** process performs the following services:

- It contacts CDS to resolve DCE pathnames (both local and foreign) that it receives from the Cache Manager. When a user on a client machine requests data that the Cache Manager has not cached, the Cache Manager employs **dfsbind** to resolve the pathname of the data. It sends **dfsbind** each element of the pathname in succession, appending each new element to the preceding elements when it sends it—for example, it first sends ***/.../*** *element_one*, then ***/.../*** *element_one****/*** *element_two*, and so on. In turn, **dfsbind** determines whether each successive pathname is valid.

  If the pathname of the data is valid, it eventually contains a DFS junction from which **dfsbind** can access information about the Fileset Database machines for the cell in which the data resides. If it encounters a junction for the DFS filespace, **dfsbind** returns information about the names and network addresses of the Fileset Database machines for the cell to the Cache Manager. (It actually decomposes binding handles to learn this information.)

  The Cache Manager uses the information from **dfsbind** to create an RPC binding that it employs to communicate with a Fileset Location (FL) Server on an appropriate Fileset Database machine. The FL Server examines the FLDB and tells the Cache Manager which File Server machine houses the fileset that contains the data requested by the user.

  For each successive pathname that it attempts to resolve for the Cache Manager, the **dfsbind** process returns one of the following error codes to the Cache Manager to indicate the result of the resolution operation:

  0 (zero)
  : Indicates that the pathname corresponds to a DFS junction that contains information about the Fileset Database machines in the cell. The process sends information about the Fileset Database machines to the Cache Manager.

  EISDIR
  : Indicates that the pathname is a valid DFS junction name prefix but is not itself a DFS junction. The process returns the valid pathname to the Cache Manager.

  ENOENT
  : Indicates that the given path could not be traversed. The process returns the invalid pathname to the Cache Manager.

  ETIMEDOUT
  : Indicates that unexpected errors occurred. The process returns only the error code to the Cache Manager.

  DCE pathname and DFS junction information that the Cache Manager receives from **dfsbind** is valid for a limited amount of time. The **dfsbind** process associates a TTL with all information it sends to the Cache Manager. The TTL defines the amount of time for which the Cache Manager is to consider the information valid. The Cache Manager caches the TTL with the information. Once its TTL has elapsed, the information becomes stale; the Cache Manager contacts **dfsbind** to refresh the information the next time it needs it.

  The **dfsbind** process associates the TTLs with the information it passes to the Cache Manager as follows:

- Information about Fileset Database machines (error code **0**) receives a TTL of 24 hours by default. (The TTL of such information can be modified with the **dfsbind** command's **-junctionlife** option.)
- Information about valid DFS junction name prefixes (error code **EISDIR**) has a TTL of 24 hours by default. (The TTL of this type of information can be changed with the command's **-prefixlife** option.)
- Information about invalid pathnames (error code **ENOENT**) has a TTL of 1 hour by default. (The TTL of this type of information can be altered with the command's **-notfoundlife** option.)

For example, when the Cache Manager first needs to access data from a fileset in the local cell, it passes each successive element of the DCE pathname of the data to **dfsbind**. If the path contains a DFS junction name, **dfsbind** eventually returns information about the local cell's Fileset Database machines, and a TTL that it assigns to the information, to the Cache Manager. The Cache Manager caches the information and the TTL, using the information to contact a Fileset Database machine in the cell. If the Cache Manager needs to access data from a fileset in the local cell before the TTL has elapsed, it uses the cached information to contact a Fileset Database machine in the cell. However, if it needs to access data from a fileset in the local cell after the TTL has elapsed, it again contacts **dfsbind** to refresh its knowledge of local Fileset Database machines.

- It obtains user authentication information for the kernel RPC runtime. It communicates with the DCE Security Service of the appropriate cell to obtain authentication information about users of the client machine.

  The Cache Manager communicates with the kernel RPC runtime when it needs to create an RPC binding to a File Server machine on behalf of a user. The kernel RPC runtime then communicates with **dfsbind** to obtain authentication information about the user for use in the binding. The **dfsbind** process obtains the authentication information from the security server and sends it back to the kernel RPC runtime, which packages the information along with the other information from the Cache Manager into the RPC binding and sends it to the appropriate File Server machine.

On a File Server machine, the **dfsbind** process simply maintains user authentication information required by the File Exporter on the machine. The File Exporter uses this information to ensure that only authenticated users access data from the machine.

The command's **-expressprocs** and **-regularprocs** options can be used to change the default number of processes **dfsbind** runs on a machine as follows:

- The **-expressprocs** option specifies the number of express processes that **dfsbind** allocates for the handling of requests that require little time to complete. For example, express processes service requests for information from the local security service. The **dfsbind** process can typically handle these types of requests more quickly than it can those assigned to regular processes.
- The **-regularprocs** option specifies the number of regular processes that **dfsbind** allocates for the handling of requests that may require a substantial amount of time to complete. For example, regular processes service requests for the resolution of DCE pathnames and for information from the security service of a foreign cell. The **dfsbind** process typically requires more time to handle these types of requests than it does to handle requests assigned to express processes.

Employing two types of processes allows **dfsbind** to function more efficiently. Requests are assigned to processes according to the amount of time they require

to complete. Thus, requests with short turnaround times are not queued behind requests with long turnaround times. Increase the number of express and regular daemons on a machine that experiences a large number of timeout (**ETIMEDOUT**) errors. (Note that both express and regular processes run as threads rather than processes, so neither type of process shows up in the output of the **ps** command or its equivalent.)

If the **-debug** option is included with the **dfsbind** command, the process provides debugging information as it executes. The debugging output is in the form of brief messages reporting the action currently being performed. The messages are useful primarily to ensure that the process is executing properly. If the **-debug** option is included with the command, the process does not automatically place itself in the background after it starts.

Privileges Required

The issuer must be **root** on the local machine.

EXAMPLES

The following line, entered in the appropriate initialization file (**/etc/rc** or its equivalent) on a client or File Server machine, starts the **dfsbind** process on the local machine. This line must be included before the line that starts the **dfsd** or **fxd** process on a client or File Server machine. The **dfsbind** process in the example uses two express processes and two regular processes.

```
dfsbind -expressprocs 2 -regularprocs 2
```

RELATED INFORMATION

Commands: **dfsd(8dfs)**, **fxd(8dfs)**.

dfsd

Purpose

Initializes the DFS Cache Manager and starts related daemons

Synopsis

dfsd[**-blocks** *number_of_cache_blocks*][**-files** *number_of_cache_files*]
[**-stat** *number_of_status_cache_entries*][**-rootfileset** *root_fileset*][**-cachedir** *cache_directory*][**-mountdir** *DFS_mount_directory*][**-rootcell** *root_cell*]
[**-settime**][**-mainprocs** *number_of_background_daemons*]
[**-tokenprocs** *number_of_token_daemons*][**-ioprocs** *number_of_I/O_background_daemons*][**-memcache**][**-dcache** *number_of_entries*][**-chunksize** *chunk_exponent*][**-namecachesize** *number_of_name_cache_entries*][**-initiallocalprotectlevel** *level*]
[**-minlocalprotectlevel** *level*][**-initialremoteprotectlevel** *level*]
[**-minremoteprotectlevel** *level*][**-anonaccess** | **-noanonaccess**][**-verbose**][**-debug**]
[**-callbackhint** *callback_IP_address_hint*][**persistentrequests** *persistent_requests_timeout_value*]
[**-help**]

OPTIONS

-blocks *number_of_cache_blocks*

Specifies the number of kilobytes to be made available for caching in the machine's cache directory (for a disk cache) or memory (for a memory cache). This value overrides the default, which must be specified in the third field of the *dcelocal***/etc/CacheInfo** file. The unit of measurement for block size is always kilobytes.

A disk cache should not exceed 85% of the disk space available on the cache partition; a memory cache should not exceed 20 to 25% of the machine's available memory. These limits are necessary because the implementation of the cache requires a small amount of disk space or machine memory, and because a memory cache must leave enough memory for processes and applications to run.

For a memory cache, do not combine this option with the **-dcache** option.

Note: The minimum cache size with the **-blocks** option for both disk and memory caching is the maximum of either 512 kilobytes or (8 * chunk size) in kilobytes. If you specify a cache size smaller than this, the cache manager readjusts the size accordingly.

-callbackhint *callback_IP_address_hint*

Specifies an IP address hint to be considered by the DFS client when selecting the callback address to be provided to the DFS fileserver during connection setup. The IP address is specified using dot notation (for example, 130.12.45.2). Normally, the DFS client attempts to choose the best address when there is more than one to choose from. The correct choice might not be made in all circumstances. The **callbackhint** provides a mechanism to bias the selection process.

This option is useful in environments where the DFS client has multiple configured network interfaces and full connectivity does not exist between the DFS fileserver and the DFS client over all the configured interfaces.

-files *number_of_cache_files*

Specifies the number of V files (chunks) to be created in the cache directory for a disk cache. This value overrides the default, which is the number of cache blocks divided by 8.

Each V file can accommodate a chunk of data. By default, each chunk can accommodate 64 kilobytes of data. To operate most efficiently, at least 85 of the cache must be in use. Use the -files option to increase the number of V files if this is not the case. Do not specify a value greater than 32,000.

Do not combine this option with the **-memcache** option, which is used for memory caching.

Note: The minimum number of V files you can specify with the **-files** option is 2. If you specify a value smaller than 2, the Cache Manager creates a cache with two V files.

-stat *number_of_status_cache_entries*
Specifies the number of entries in the machine's memory for recording status information about DFS files in the cache. The default is **300**.

-rootfileset *root_fileset*
Names the read/write fileset corresponding to the top-level (**root**) directory. This option is generally used for testing purposes only.

-cachedir *cache_directory*
Names the local disk directory to be used as the cache for disk caching. This value overrides the default, which must be specified in the second field of the **CacheInfo** file. The default is *dcelocal***/var/adm/dfs/cache**.

Do not combine this option with the **-memcache** option, which is used for memory caching. With memory caching, the **-cachedir** option, like the second field of the **CacheInfo** file, is ignored.

-mountdir *DFS_mount_directory*
Names the local disk directory where the DCE global namespace is to be mounted. This value overrides the default, which must be specified in the first field of the **CacheInfo** file. The default for a machine with a disk is the global namespace designation (***/...***); if ***/...*** is not used, symbolic links to the global namespace will not work.

-rootcell *root_cell*
Names the cell that contains the root fileset. This option is generally used for testing purposes only.

-settime
Causes the local machine to select a random server machine in the local cell to use as the source of the correct time. If this option is specified, the local machine selects a server machine and checks the time on that machine every 10 minutes. If the time on the local machine differs by more than 2 seconds from the time on the selected server machine, the local machine adjusts its time to match that of the server machine.

For machines running the DCE Distributed Time Service (DTS) or the Network Time Protocol (NTP), it is recommended that the **-settime** option be omitted to prevent the machine from selecting and using two different time standards at once.

-mainprocs *number_of_background_daemons*
Specifies the number of background daemons to run on the machine. These daemons improve efficiency by performing prefetching and background writing of saved data. The default is two.

Increase the number of background daemons if the machine serves more than five users.

-tokenprocs *number_of_token_daemons*

Specifies the number of background daemons dedicated to servicing token revocation RPC requests from File Exporters. The default is two. (Token daemons run in addition to the background daemons associated with the **-mainprocs** option.)

Increase the number of token daemons if users on this machine interact with many File Server machines.

-ioprocs *number_of_I/O_background_daemons*

This option is ignored on machines running AIX.

-memcache

Causes **dfsd** to initialize a memory cache rather than a disk cache. If this option is provided, space in memory is allocated for the cache; no disk space is used, even if it is available.

Do not combine this option with the **-files** option (which is used for machines that use disk caching). Also, do not combine this option with the **-cachedir** option; with memory caching, the **-cachedir** option, like the second field of the **CacheInfo** file, is ignored.

-dcache *number_of_entries*

Sets the number of dcache entries in memory; dcache entries store information about cache chunks.

For a disk cache, the *dcelocal***/var/adm/dfs/cache/CacheItems** file contains one entry for each V file. By default, 100 entries from the **CacheItems** file are duplicated in machine memory; the **-dcache** option overrides the default.

For a memory cache, there is no **CacheItems** file; one dcache entry exists for each cache chunk. The Cache Manager determines the number of dcache entries (cache chunks) by dividing the cache size by the chunk size; the **-dcache** option sets the number of cache chunks. Do not combine this option with the **-blocks** option.

Use of this option with a disk cache is not necessary because it increases performance only marginally. It is not recommended with a memory cache because it requires the issuer to perform additional calculations.

-chunksize *chunk_exponent*

Sets the size of each cache chunk. Provide an integer between 13 and 18 to be used as an exponent of 2. This value overrides the default chunk size, which is 64 kilobytes (2^{16}) for a disk cache and 8 kilobytes (2^{13}) for a memory cache. A value less than 13 or greater than 18 sets the chunk size to the appropriate default for the type of cache in use. The unit of measure for chunk size is always bytes.

It is not recommended that you use this option with the **-dcache** option for a memory cache.

-namecachesize *number_of_name_cache_entries*

Sets the number of entries allocated for the Cache Manager's name lookup cache. Provide an integer greater than 0 (zero); the default number of name cache entries is **256**.

The name lookup cache stores the results obtained from remote directory lookup requests to DFS servers, which allows subsequent lookup requests for the same file or directory to be satisfied on the local DFS client rather than on the remote DFS server. Because name cache entries are recycled

when the name lookup cache limit is reached, the ability to satisfy the request locally depends upon the size of the name lookup cache.

-initiallocalprotectlevel *level*
Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-minlocalprotectlevel *level*
Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-initialremoteprotectlevel *level*
Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-minremoteprotectlevel *level*
Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-anonaccess
Allows access to untrusted cells.

-noanonaccess
Does not allow access to untrusted cells.

-verbose
Directs **dfsd** to produce a more detailed trace of its activities than it does by default. The trace is displayed on standard output (**stdout**) unless it is directed elsewhere.

-debug
Causes **dfsd** to produce a highly detailed trace of its activities, which can be useful for debugging purposes. The trace is displayed on standard output (**stdout**) unless it is directed elsewhere.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with **dfsd**. See the **bos_help(8dfs)** and **bos_apropos(8dfs)** reference pages for examples using these commands.

-persistentrequests *persistent_requests_timeout_value*
Specifies the amount of time in seconds the Cache Manager is to continue trying a request before returning an error due to irregular hardware or software outages preventing communication with the DFS file exporter.

A value of zero (0) will assign the default of 86,400 seconds (24 hours) as a timeout value.

DESCRIPTION

The **dfsd** process initializes the DFS Cache Manager on a client machine according to the information specified with the options described previously. It must be run on all DFS client machines. It is usually added to the proper start-up file (**/etc/rc** or its equivalent) rather than typed at the command shell prompt. (The **dfsbind** process must be run before the **dfsd** process in a start-up file.) The binary file for the **dfsd** process resides in *dcelocal***/bin/dfsd**.

Specifically, the **dfsd** process does the following:

- Transfers information about cell membership to kernel memory. This information can be changed only by rebooting and running **dfsd**.
- Determines if the cache is on the local disk or in machine memory. A disk cache is used unless the **-memcache** option is provided. If the **-memcache** option is used, no disk space is used, even if it is available; the Cache Manager maintains all cached data and cache-related information in memory.
- Defines the name of the local disk directory devoted to a disk cache. The second field in the **CacheInfo** file specifies the default directory. If necessary, **dfsd** creates the directory, provided its parent directory exists. Any directory that formerly served as the disk cache is left on the disk.
- Sets the size of the cache. The third field in the **CacheInfo** file specifies the default cache size in kilobytes.

  For a disk cache, the value in the **CacheInfo** file is an upper limit that can be increased only with the **-blocks** option; it cannot be increased with the other options available with the **dfsd** process. For a memory cache, the **-dcache** option alone or in combination with the **-chunksize** option overrides the cache size specified in the **CacheInfo** file; these combinations are not recommended.

  After initialization, use the **cm setcachesize** command to change the size of a disk cache without rebooting. The value set with the **cm setcachesize** command is overridden the next time the machine is rebooted and **dfsd** is run. The **cm setcachesize** command does not work for memory caches; the machine must be rebooted. (The **cm getcachesize** command can be used to display the current size of the cache, the amount in use, and the type of cache—disk or memory.)
- Sets the size of each chunk of data in the cache and, by implication, the amount of data the Cache Manager requests at one time from the File Exporter. For a memory cache, if the total cache size divided by the chunk size leaves a remainder, **dfsd** rounds the number down, resulting in a slightly smaller cache.
- Sets the number of dcache entries allocated in machine memory for storing information about the cache chunks in a disk cache.
- Sets the number of empty V files created in the cache directory for a disk cache. (A memory cache cannot use V files because it does not use disk storage; the number of chunks is instead equal to the number of dcache entries.)
- Sets the number of **stat** entries in machine memory for caching status information about cached DFS files.
- Sets the number of entries in the name lookup cache for storing the results of remote directory lookups.
- Specifies the directory on the machine's local disk where DFS is mounted. The first field in the **CacheInfo** file specifies the default directory.

- Selects a random server machine in the local cell as the source of the correct time if the **-settime** option is provided.
- Sets the initial RPC authentication level and minimum RPC authentication bound for communications between the Cache Manager and File Servers.
- Sets the timeout value for the Cache Manager to use for **persistentrequests**.

In addition to setting cache configuration parameters, **dfsd** also starts the following types of daemons. On most system types, these daemons appear as nameless entries in the output of the **ps** command.

- One or more maintenance daemons, which perform routine periodic maintenance tasks such as the following:
  - Performing garbage collection
  - Synchronizing files
  - Probing processes on File Server machines every few minutes
  - Refreshing information about filesets referenced by the Cache Manager once per hour
  - Keeping the machine's clock synchronized with the clock of the chosen server machine (if the **-settime** option is included with the **dfsd** command)
- One or more background daemons, which improve performance by performing delayed writing of updated data. The default number of background daemons is two, which is usually sufficient to handle up to five simultaneous users of a machine. Use the **-mainprocs** option to increase the number of background daemons if the machine serves more than five users.
- One or more token daemons, which handle token revocation RPC requests from the File Exporters on File Server machines (for example, by writing modified data back to the File Server machines). The default number of token daemons is two. Use the **-tokenprocs** option to increase this number if the machine interacts with many File Server machines from different cells.

The default number of daemons is ten (one maintenance daemon, two background daemons, two token daemons, and five I/O daemons). You can alter only the number of background daemons, token daemons, and I/O daemons; **dfsd** initializes additional maintenance daemons as necessary.

RPC Security Settings

The **dfsd** command sets the DCE RPC security level for RPCs sent between a Cache Manager and DFS File Servers. The command sets two levels: an initial DCE RPC security level used as a starting point in security level negotiations between the Cache Manager and a File Server, and the minimum DCE RPC security level that the Cache Manager will accept for such communications. Two sets of these levels are maintained: One set specifies the security levels for communications with File Servers within the local cell, and the other set specifies the security levels for communications with File Servers within foreign cells. Both sets of security levels can be adjusted through the **cm setprotectlevels** command.

In operation, the Cache Manager and File Server interact to arrive at a mutually acceptable authentication level for communications. The negotiation starts with an RPC that uses the initial authentication level sent from the Cache Manager to the File Server. If the initial authentication level is outside the minimum or maximum bounds set at the File Server, the File Server returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts

its RPCs to an acceptable security level or the File Server requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Server). Once the Cache Manager and File Server have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Server.

The Cache Manager and **fxd** default settings are such that communications occur at the packet authentication level for local cell and packet integrity authentication level for non-local cells.

In addition to a general pair of upper and lower bounds for all communications between the File Server and Cache Manager, administrators can also set advisory bounds on a per-fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level (they may be enforced in a future version of DFS). Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset.

The various authentication levels are set by specifying either an integer value between 0 and 6, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **rpc_protect_level_default** or **default** or **0**: Use the DCE default authentication level.
- **rpc_protect_level_none** or **none** or **1**: Perform no authentication.
- **rpc_protect_level_connect** or **connect** or **2**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_protect_level_call** or **call** or **3**: Authenticate only at the beginning of each RPC received.
- **rpc_protect_level_pkt** or **pkt** or **4**: Ensure that all data received is from the expected host.
- **rpc_protect_level_pkt_integrity** or **pkt_integrity** or **5**: Authenticate and verify that none of the data transferred has been modified.
- **rpc_protect_level_pkt_privacy** or **pkt_privacy** or **6**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

Privileges Required

The issuer must be **root** on the local machine.

EXAMPLES

It is recommended that the **dfsd** process be included in the proper initialization file (**/etc/rc** or its equivalent) rather than typed at the command shell prompt. The **dfsbind** process must be run before the **dfsd** process in a start-up file. For most disk caches, the following form is appropriate in the initialization file:

```
dcelocal/bin/dfsd
```

The following line in an initialization file is appropriate when enabling a machine to serve more than five users:

```
dcelocal/bin/dfsd -mainprocs 4
```

The following line in an initialization file initializes a memory cache and sets the chunk size to 16 kilobytes (2^{14}):

```
dcelocal/bin/dfsd -memcache -chunksize14
```

RELATED INFORMATION

Commands: **cm getcachesize(8dfs)**, **cm getprotectlevels(8dfs)**, **cm setcachesize(8dfs)**, **cm setprotectlevels(8dfs)**, **dfsbind(8dfs)**, **fts setprotectlevels**.

Files: **CacheInfo(4dfs)**, **CacheItems(4dfs)**, **FilesetItems(4dfs)**, **Vn(4dfs)**.

dfsexport

Purpose

Exports DCE LFS aggregates and non-LFS partitions to the DCE namespace

Synopsis

dfsexport{**-all -aggregate** *name*}[**-type** *name*][**-detach**][**-force**]
[**-verbose][-help**]

OPTIONS

-all Specifies that all aggregates and partitions listed in the *dcelocal***/var/dfs/dfstab** file are to be exported. Use the **-type** option with this option to export only DCE LFS aggregates or only non-LFS partitions. Use this option or use **-aggregate** ; omit both options to list all aggregates and partitions currently exported from the local disk to the DCE namespace.

-aggregate *name*
Specifies the device name or aggregate name of the aggregate or partition to be exported. These names are specified in the first and second fields of the entry for the aggregate or partition in the **dfstab** file. Use this option or use **-all** ; omit both options to list all aggregates and partitions currently exported from the local disk to the DCE namespace.

-type *name*
Used with the **-all** option, specifies that only aggregates or partitions whose file system types match the type specified with this option are to be exported. The type can be specified as **lfs** to export only DCE LFS aggregates, or it can be specified as **ufs** to export only non-LFS partitions. The type of each aggregate or partition appears in the third field of the entry for the device in the **dfstab** file. The type must be specified in lowercase letters (as it appears in the **dfstab** file).

Use this option only with the **-all** option; it is ignored if it is used without the **-all** option. If it is omitted and **-all** is used, the command exports both **lfs** and **ufs** devices.

-detach
Specifies that the aggregates or partitions indicated with the command's other options are to be detached (no longer exported), making them unavailable via the DCE namespace. Use **-all** or **-aggregate** with this option to indicate the devices to be detached; use the **-type** option with **-all** to detach only one type of device.

Use the **-detach** option only when no users are accessing data on the aggregate or partition to be detached or when a serious emergency warrants its use. When the **-detach** option is used, the command revokes all tokens for data on a device before it detaches it. It does not detach a device unless it can revoke all necessary tokens. You can use the **-force** option to direct the command to detach a device even if it cannot revoke all necessary tokens.

To permanently detach an aggregate or partition, it must also be removed from the **dfstab** file. Otherwise, the **dfsexport** command exports the aggregate or partition the next time it is run (provided the aggregate or partition is included in the specification for the devices to be exported).

-force Used with the **-detach** option, directs the **dfsexport** command to detach an aggregate or partition even if it cannot revoke all tokens for data on the device. By default, the command does not detach a device unless it can revoke all necessary tokens. Use this option only when a serious emergency requires its use.

This option can be used only with the **-detach** option. The command fails if this option is used with any combination of options that does not include the **-detach** option.

-verbose
Directs the command to report on its actions as it executes.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **dfsexport** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The **dfsexport** command exports DCE LFS aggregates and non-LFS disk partitions from the local disk of a machine to the DCE namespace. File systems on exported aggregates and partitions are available to other users in the DCE namespace. The binary file for the **dfsexport** command resides in *dcelocal***/bin/dfsexport**.

The command exports DCE LFS aggregates, non-LFS partitions, or both, based on the values provided with its options. If the **-all** option is provided, the command exports all aggregates and partitions listed in the *dcelocal***/var/dfs/dfstab** file. If the **-aggregate** option is provided, it exports only the aggregate or partition whose device name or aggregate name is specified with the option. The specified name must be listed in the **dfstab** file.

The **-type** option can be used with the **-all** option to indicate that only DCE LFS aggregates or only non-LFS partitions are to be exported. If **lfs** is provided with the **-type** option, the command exports only DCE LFS aggregates; if **ufs** is specified with the **-type** option, it exports only non-LFS partitions. If the **-type** option is used, the **-all** option must also be included; otherwise, the **-type** option is ignored.

When **dfsexport** executes, it reads the **dfstab** file on the local disk of the machine to determine the aggregates and partitions available to be exported. An aggregate or partition must have an entry in the **dfstab** file if it is to be exported. Because this command reads the **dfstab** file, information supplied with its options must match exactly the information for an aggregate or partition specified in that file.

The **dfsexport** command reads a list of all currently exported aggregates and partitions that is maintained in the kernel of the local machine. The command will not export an aggregate or partition that is currently exported. The command also refuses to export a DCE LFS aggregate that needs to be recovered with the **salvage** command. If the **dfsexport** command fails with an exit status of **2**, use the **salvage** command to recover the aggregate that caused the failure and reissue the **dfsexport** command.

Issuing the **dfsexport** command with no options lists the aggregates and partitions currently exported from the local disk to the DCE namespace. The **fts lsaggr** command can also be used to display a current list of all aggregates and partitions exported from a machine.

The **dfsexport** command is generally included in a machine's initialization file (**/etc/rc** or its equivalent) rather than issued at the keyboard. Once included in the initialization file, the command automatically exports all indicated aggregates and partitions whenever the machine is rebooted. Typically, the command is included with its **-all** option to export all aggregates and partitions listed in the **dfstab** file.

Prior to using this command to export a non-LFS partition for the first time, perform the following steps:

1. Ensure that the partition is mounted locally; it can contain data or it can be empty.
2. Issue the **fts crfldbentry** command to register the non-LFS fileset that resides on the partition (each non-LFS partition contains a single fileset) in the Fileset Location Database (FLDB). The Fileset Location Server (FL Server) can then track the fileset's location. The **fts crfldbentry** command also allocates a unique fileset ID number for the non-LFS fileset.
3. Create an entry for the non-LFS partition in the **dfstab** file on the machine on which the partition resides. Use the aggregate ID number specified with the **-aggrid** option of the **fts crfldbentry** command and the fileset ID number allocated by the command in the fourth and fifth fields of the entry for the partition. Also, use the name of the partition's local mount point as its aggregate name in the second field of its entry. (Once these steps are complete, use the **fts crmount** command to mount the non-LFS fileset that resides on the partition.)

   Note: On DCE 2.2 for AIX, steps 2 and 3, above, can be done in one step with the **mkfilesys.dfs** command or by using the AIX SMIT utility.

Before exporting a non-LFS partition, also make sure that no users have files open on the partition. DFS cannot effectively synchronize file access between users who opened files from a non-LFS partition before the partition was exported and users who open files from the partition after the partition is exported because only the latter have tokens.

Before using this command to export a DCE LFS aggregate for the first time, complete the following steps:

1. Ensure that the disk partition on which the aggregate is to reside is initialized with the **newaggr** command; the partition cannot contain data when the **newaggr** command is executed. The **newaggr** command needs to be run on a partition only once. *Do not use the***newaggr** *command to reinitialize a partition that contains data you want to preserve; the command destroys any data on the partition on which it is used.*
2. Create an entry for the DCE LFS aggregate in the **dfstab** file on the machine on which the aggregate is located. (Once the aggregate is exported, the **fts create** command can be used to create and register filesets on the aggregate, after which the **fts crmount** command can be used to mount the new filesets.)

   Note: On DCE 2.2 for AIX, step 2 above can be done with the **mkfilesys.dfs** command or by using the AIX SMIT utility.

The **dfsexport** command can also be used to detach an exported aggregate or partition from the DCE namespace. Detaching an aggregate or partition makes it unavailable in the namespace. To detach one or more aggregates or partitions, use the **-all** (and optionally the **-type**) option or the **-aggregate** option to specify the devices to be detached, and include the **-detach** option with the command.

Before it detaches a device, the command revokes all tokens for data on the device. When their tokens are revoked, clients flush data cached from the device, writing any modified data back to the device. If the command cannot revoke all necessary tokens, it does not detach the device. (It instead displays a message reporting that the device is busy.)

The **-force** option can be used with the **-detach** option to direct the command to detach a device even if it cannot revoke all necessary tokens (that is, even if files from the device are still open). (You can also remove an aggregate or partition from the DCE namespace by removing its entry from the **dfstab** file and rebooting the machine.)

Note: When AIX JFS file systems such as **/tmp** are exported into the DFS file space, subsequent **dfsexport -detach** commands might fail to detach these file systems. This failure is caused by local activity occurring on the exported file system at the same time the detach is being attempted.

Privilege Required

If the command is issued with no options to list the aggregates and partitions exported from the local machine, no privileges are required. Otherwise, the issuer must be logged in as **root** on the local machine.

CAUTIONS

Before detaching an aggregate or partition, attempt to ensure that no users are currently accessing data from filesets on the device. The command revokes all tokens for data on the device before it detaches it, which causes clients to flush data cached from the device (writing any modified data back to the File Server machine). However, a user who is accessing data from the device will no longer be able to save the data. Any attempt to perform an action that involves a detached aggregate or partition elicits a message reporting that the device is unknown. Exercise special caution before using both the **-detach** and **-force** options, which forces a device to be detached even if all tokens cannot be revoked (that is, even if files are still open).

EXAMPLES

The following command line is typically added to a machine's initialization file (**/etc/rc** or its equivalent). The line exports all of the aggregates and partitions that have entries in the machine's **dfstab** file.

```
dfsexport -all
```

The following command exports the aggregate whose device name (as it appears in the **dfstab** file) is **/dev/lv02**:

```
# dfsexport /dev/lv02
```

The command that follows exports all DCE LFS aggregates (all entries in the **dfstab** file with file system type **lfs**):

```
# dfsexport -all -type lfs
```

The **dfsexport** command can return the following exit values:

0 The command completed successfully.

1 The command failed for a reason other than that associated with an exit value of **2**.

2 The command failed because a DCE LFS aggregate to be exported needs to be recovered with the **salvage** command before it can be exported.

RELATED INFORMATION

Commands: **fts create(8dfs)**, **fts crfldbentry(8dfs)**, **fts crmount(8dfs)**, **fts lsaggr(8dfs)**, **newaggr(8dfs)**, **salvage(8dfs)**.

Files: **dfstab(4dfs)**.

dfsiauth

Purpose

Performs operations against the NFS/DFS Translator to authenticate a DCE principal; register, unregister, or list host/uid pairs for translation, and optionally associate @sys and @host values for path substitution.

Synopsis

```
dfsiauth -add [overwrite] | -delete -r  remote_host -i remote_uid[-u principal]
[-p password ][-s sysname ][-h hostname ]
dfsiauth -list[-u principal ][-p password ] | -flush
```

OPTIONS

-r *remote_host*
Specifies the hostname of the user requesting authenticated access.

-l *remote_uid*
Specifies the uid of the user requesting authenticated access.

-u *principal*
Specifies the DCE principal with which to authenticate.

-p *password*
Specifies the password of the DCE principal.

-s *sysname*
Associates the parameter sysname with the @sys value for the input host/uid pair.

-h *hostname*
Associates the parameter hostname with the @host value for the input host/uid pair.

-add Adds the specified mapping information.

-delete
Deletes the specified mapping information.

-overwrite
Changes information about an existing mapping. This option is only valid with the **-add** option.

-list Lists the registered authentication mappings.

-flush Removes expired authentication mappings.

DESCRIPTION

The **dfsiauth** command performs operations against the NFS/DFS Translator for a remote host, and remote uid with an existing DCE principal. It is issued on the machine that performs NFS/DFS translation. The DCE principal's password must be included. The password can be included on the command line (**-p** *password*). When the password is not included, the command prompts for the password.

Options such as **-add** and **-delete** might be abbreviated on the command line to **-a**, **-d**, or other shortened forms of the option. Specific option information includes:

- The **-add** option is used to register a hostname and uid pair to a DCE principal. Along with performing an association for NFS service, the command can also be used to associate and @sys and @host name for path element substitution. If a principal is not specified, an unauthenticated mapping is assumed and only @sys and @host values are registered. No attempt is made to validate the selected values of @sys and @host.
- The **-overwrite** option is used only with the **-add** option to overwrite an existing authentication mapping. This option is useful only if the @sys value needs to be changed because of a software upgrade. In order to overwrite existing information, the principal that initially set up the mapping must also be given as the principal requesting to overwrite the information.
- The **-delete** option is used to remove an authenticating mapping from the NFS/DFS Translator. The principal that added the mapping must be the same as the principal who is attempting to delete the mapping. This information is checked before the deletion is attempted. If the principals do not match, an error is returned. The principal is not checked when attempting to remove unauthenticated mappings, for example @sys and @host mappings. Also, a mapping cannot be deleted if the principal has been removed from the DCE registry.
- The **-list** option displays the hostname and uid pairs that are registered with the NFS/DFS Translator. A user must provide the DCE principal and password before the registered mappings can be displayed. Only the user's mapping are displayed. The local root user may issue this command.

The NFS/DFS Translator periodically performs automatic removal of expired mappings. However, the **dfsiauth -flush** option may also be used to manually remove all expired and unauthenticated authentication mappings from the NFS/DFS Translator. Only the local root user may issue this command.

Before using **dfsiauth**, the **startnfs.dfs** command must be run. The **startnfs.dfs** command starts the NFS to DFS authenticating Translator. It makes sure that all required daemons are running, and then loads the kernel extension.

The following daemons must be running in order for NFS to DFS authenticating to function properly:

- **dfsd**
- **nfsd**
- **rpc.statd**
- **rpc.lockd**
- **rpc.maintd**
- **portmap**

Privilege Required

This program runs as a setuid program. Issuer must be the local root user in order to execute the **-flush** option.

EXAMPLES

Adding a Mapping With the Password on a Command Line

The following command and system response authenticates to DCE as the principal, *dawn*, and registers the remote host, *sunlight*, and remote uid, *1065*. The sysname, **rs_aix41**, is also associated with this authentication mapping.

```
$ dfsiauth -add -r sunlight -i 1065 -u dawn -p -dce- -s rs_aix41
dfsiauth: <sunlight, 1065> mapping added
DCE principal: dawn
System Type (@sys) rs_aix41
```

Adding a Mapping When Prompted for a Password

The following example and system response shows the authentication mapping for remote host, **spooky.austin.ibm.com**, and **uid**, 654, where the command prompts for the password.

```
$ dfsiauth -ass -r spooky.austin.ibm.com -i 654 -u ghost
Password:  boo
dfsiauth: <spooky.austin.ibm.com,654> mapping added
DCE principal: ghost
```

Adding a Mapping With @sys and @host Values

The following example and system response shows setting the @sys value, *intel_os2*, and @host value, mystic, to be associated with the remote host, *mystic*, and remote uid, *1022*. This mapping sets up @sys and @host values for an unauthenticated DFS user.

```
$ dfsiauth -add -r mystic -i 1022 -s intel_os2 -h mystic
dfsiauth: <mystic,1022> mapping added
System Type (@sys) intel_os2, Host Type (@host) mystic
```

Changing an Existing Mapping

The following example and system response overwrites the existing mapping of remote host, *sunlight*, remote uid, *1065*, by changing the current rs_aix41 @sys value with the value, *sun*. This mapping was previously registered using the DCE principal, *dawn*.

```
$ dfsiauth -add -overwrite -r sunlight -i 1065 -u dawn -p -dce- -s sun
dfsiauth: <sunlight,1065> mapping changed
DCE principal: dawn
System Type (@sys) sun
```

Deleting an Existing Mapping

The following example and system response shows deletion of the authentication mapping for the remote host, spooky.austin.ibm.com, and remote uid, *654*, associated with DCE principal, *ghost*.

```
$ dfsiauth -delete -r spooky.austin.ibm.com -i 654 -u -p boo
dfsiauth: <spooky,austin.ibm.com, 654> mapping deleted
DCE principal: ghost
```

Displaying Existing Mappings for All Users

The following example and system response lists the registered authentication mappings. The user is logged in as the local root user.

```
$ dfsiauth -list
Host             Uid       Principal     @sys  Expiration
----             ---       ---------     ----  ----------
sunlight.austin  1065      dawn                 2/27/98  12:00
```

RELATED INFORMATION

Commands: **startnfs.dfs**

dfstrace

Purpose

Introduction to the **dfstrace** command suite

OPTIONS

The following options are used with many **dfstrace** commands. They are also listed with the commands that use them.

-set *set_name*
Specifies the name of an event set to be utilized by the command. An event set is a module designed to track specific events within the DFS kernel or within one or more server processes. Each event set generates trace messages relative to the type of events that it tracks and writes information on each event to, from one to eight trace logs. Tracing by event set allows users of the **dfstrace** commands to more quickly isolate and diagnose a problem.

Following are some of the DFS kernel event sets that you can see:

- **cm** − Cache Manager package
- **fshost** − File exporter host package
- **fx** − File exporter package
- **episode/anode** − LFS anode package
- **episode/logbuf** − LFS buffer/logging package
- **episode/vnops** − LFS vnode package
- **tkc** − Token cache package
- **tkm** − Token manager package
- **tpq** − Thread pool queue package
- **xops** − Vnode-to-fileset synchronization package

Following are some of the server process event sets that you can see:

- **bosserver** − **bosserver** package
- **dacl** − DFS ACL package
- **dfsauth** − DFS security package
- **flserver** − **flserver** package
- **ftserver** − **ftserver** package
- **ftutil** − Fileset utility package
- **ubikdisk** − Disk I/O subset of Ubik package
- **ubikvote** − Sync site election subset of Ubik package

-log *log_name*
Specifies the name of a server process or kernel trace log to be utilized by the command. All logs are stored in kernel or server process memory that is allocated on the initialization of the kernel or server process. The default size of a log, which is measured in 4-kilobyte units (kwords), is predefined; however, this size can be changed by a system administrator. Any number of event sets can write to a single log.

-cdsentry *server_entry_in_CDS*
Specifies the name of a server process to which to connect. This option is

required when performing operations on server process logs and event sets; it must be omitted when performing operations on kernel logs and event sets. The full DCE pathname of a server process must be specified with **/.:/hosts/** *machine***/** *process_name*.

-help Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see the **dfs_intro(8dfs)** reference page.

DESCRIPTION

Commands in the **dfstrace** command suite are used by system administrators to diagnose problems within the DFS kernel or within the server processes that interface with the **dfstrace** command suite. This diagnosis is performed by reading the output of trace logs containing diagnostic messages written by event sets that track specific actions performed by the DFS kernel or a server process.

The commands in the **dfstrace** command suite allow you to perform the following functions:

- List information about event sets (using the **dfstrace lsset** command)
- Set an event set's state (using the **dfstrace setset** command)
- List information about trace logs (using the **dfstrace lslog** command)
- Change the size of trace logs (using the **dfstrace setlog** command)
- Dump the contents of trace logs (using the **dfstrace dump** command)
- Clear trace logs (using the **dfstrace clear** command)

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ man dfstrace
Displays the reference page for the command suite.

$ man dfstrace_ *command*
Displays the reference page for an individual command. You must use an _ (underscore) to connect the command suite to the command name. *Do not use the underscore when issuing the command in DFS.*

$ dfstrace help
Displays a list of commands in a command suite.

$ dfstrace help *command*
Displays the syntax for a single command.

$ dfstrace apropos -topic *string*
Displays a short description of any commands that match the specified *string*.

Consult the **dfs_intro(8dfs)** reference page for complete information about the DFS help facilities.

Privilege Required

Except for the **dfstrace help** and **dfstrace apropos** commands, which require no privilege, executing the **dfstrace** commands require one of the following two types of privilege, depending on the operation being performed:

- If the **dfstrace** command is executed on a kernel log or event set, the issuer must be logged in as **root** on the local machine.
- If the **dfstrace** command is executed on a server process log or event set, the issuer must be listed in the admin list associated with that process on the machine specified with the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

Specific privilege information is listed with each command's description. In addition, if the BOS Server, on the same machine as a server process, is running with DFS authorization checking disabled, no privilege is required to issue **dfstrace** commands on the event sets and logs associated with that server process.

RELATED INFORMATION

Commands: **dfs_intro(8dfs)**, **dfstrace apropos(8dfs)**, **dfstrace clear(8dfs)**, **dfstrace dump(8dfs)**, **dfstrace help(8dfs)**, **dfstrace lslog(8dfs)**, **dfstrace lsset(8dfs)**, **dfstrace setlog(8dfs)**, **dfstrace setset(8dfs)**.

dfstrace apropos

Purpose

Shows each help entry containing a specified string

Synopsis

dfstrace apropos -topic *string*[**-help**]

OPTIONS

-topic *string*
: Specifies the keyword string for which to search. If it is more than a single word, surround the string with ″ ″ (double quotes) or other delimiters. Type all strings for **dfstrace** commands in lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace apropos** command displays the first line of the help entry for any **dfstrace** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **dfstrace help** command.

Privilege Required

No privileges are required.

OUTPUT

The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **dfstrace** command where the string specified by **-topic** is part of the command name or the first line.

EXAMPLES

The following command lists all **dfstrace** commands that have the word **list** in their names or short descriptions:

```
$ dfstrace apropos list


lslog: list available logs
lsset: list available event sets
```

RELATED INFORMATION

Commands: **dfstrace help(8dfs)**.

dfstrace clear

Purpose

Clears server process or kernel trace logs

Synopsis

dfstrace clear{**-set** *set_name***-log** *log_name*}[**-cdsentry** *server_entry_in_CDS*]
[**-help**]

OPTIONS

-set *set_name*
: Specifies the name of each event set whose logs you want to clear. Specify this option or the **-log** option; omit both to clear all nonpersistent kernel logs on the local machine or all nonpersistent server process logs for the server process specified with the **-cdsentry** option.

-log *log_name*
: Specifies the name of each log that you want to clear. Specify this option or the **-set** option; omit both to clear all nonpersistent kernel logs on the local machine or all nonpersistent server process logs for the server process specified with the **-cdsentry** option.

-cdsentry *server_entry_in_CDS*
: Specifies the full DCE pathname (***/.:/hosts/*** *machine**/*** *process_name*) of a server process whose logs you want to clear. Use the **-set** or **-log** option with this option to specify a distinct group of server process logs to clear; use this option alone to clear all nonpersistent logs associated with the server process. Omit this option to clear kernel logs.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace clear** command clears the specified server process or kernel logs. If you want to clear a kernel log, it must reside on the local machine. If you want to clear a server process log, it can reside on any machine; however, you must use the **-cdsentry** option to specify the appropriate server process.

To clear a specific log, identify the log with the **-set** or **-log** option. Use the **-cdsentry** option to clear a specific server process log; omit that option to clear a specific kernel log.

To clear all kernel logs on a machine, run the **dfstrace clear** command without any options. To clear all server process logs associated with a particular server, run the command with the **-cdsentry** option only. Note that you cannot clear persistent logs in this global manner. The persistent attribute prevents accidental clearing of important logs. The attribute is assigned to a log when the kernel or server process is compiled and cannot be changed.

Privilege Required

To clear a kernel log, the issuer must be logged in as **root** on the local machine. To clear a server process log, the issuer must be listed in the **admin** list associated

with that process on the machine specified with the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

EXAMPLES

The following command clears all logs used by the **fx** kernel event set on the local machine:

```
# dfstrace clear fx
```

The following command clears all logs used by the **ftserver** process on the machine **dewitt**:

```
$ dfstrace clear -cdsentry /.:/hosts/dewitt/ftserver
```

RELATED INFORMATION

Commands: **dfstrace lslog(8dfs)**, **dfstrace lsset(8dfs)**.

dfstrace dump

Purpose

Dumps server process or kernel trace logs

Synopsis

dfstrace dump{**-set** *set_name***-follow** *log_name*}[**-file** *output_filename*] [**-sleep** *seconds_between_reads*][**-cdsentry** *server_entry_in_CDS*][**-help**]

OPTIONS

-set *set_name*
Specifies the name of each event set whose corresponding logs you want to dump. Specify this option or the **-follow** option; omit both to dump all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option. If you specify multiple event sets that point to the same log, that log is dumped multiple times.

-follow *log_name*
Specifies the name of a kernel log to continuously dump. Process server logs cannot be continuously dumped. When a log is continuously dumped, it is also cleared. Specify this option or the **-set** option; omit both to dump all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option.

-file *output_filename*
Indicates the name of a file to which to write the output of the command. If the log being dumped is a server process log, the *output_filename* cannot contain **/** (slashes); the file is automatically placed in the directory */dce/local/***var/dfs/adm**. Furthermore, if an *output_filename* is not provided, the output is placed in the file **icl.** *server_process_name*. Server process logs cannot be directly dumped to standard output. (If the output file for a server process log already exists, the older version is moved to the file *output_filename***.old**.) If the log being dumped is a kernel log, the *output_filename* must specify the full or relative pathname of the output file.

-sleep *seconds_between_reads*
Defines the number of seconds that the command pauses between dumps when dumping a kernel log in continuous mode. This option can only be used with the **-follow** option. The default value is 10 seconds.

-cdsentry *server_entry_in_CDS*
Specifies the full DCE pathname (**/.:/hosts/** *machine***/** *process_name*) of a server process whose logs you want to dump. Use the **-set** option with this option to specify a distinct group of server process logs to dump; use this option alone to dump all logs associated with the specified server process. Omit this option to dump kernel logs.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace dump** command dumps the specified kernel logs to standard output or the specified server process logs to the *output_filename* specified with the **-file** option. Server process logs cannot be directly dumped to standard output. If an

output_filename is not provided for a server process log dump, the output is place in the file **icl.** *server_entry_in_CDS*. The contents of a kernel log dump can be directed into a file, rather than to standard output, by using the **-file** option.

If you want to dump a kernel log, it must reside on the local machine. If you want to dump a server process log, it can reside on any machine; however, you must use the **-cdsentry** option to specify the appropriate server process.

To dump specific logs, identify the logs with the **-set** option. Use the **-cdsentry** option to dump specific server process logs; omit that option to dump specific kernel logs.

To continuously dump a single kernel log, issue the command with the **-follow** option. Server process logs cannot be continuously dumped.

To dump all kernel logs on a machine, run the **dfstrace dump** command without the **-set** or **-follow** option. To dump all server process logs associated with a particular server, run the command with the **-cdsentry** option, but without the **-set** option.

Privilege Required

To dump a kernel log, the issuer must be logged in as **root** on the local machine. To dump a server process log, the issuer must be listed in the **admin** list associated with that process on the machine specified by the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

OUTPUT

At the beginning of the output of each dump is the date and time at which the dump began. Unless the **-follow** option is specified, the number of logs being dumped is displayed. The content of each log is preceded by a message identifying the log.

Each log message contains the following three components:

- The time stamp associated with the message
- The process ID or thread ID associated with the message
- The message itself

Every 1024 seconds, a current time message is written to each log. This message has the following format:

```
time timestamp, pid 0: Current time: unix_time
```

Use the current time message to determine the actual time associated with each log message as follows:

1. Locate the log message whose actual time you want to determine.
2. Search backward through the dump record until you come to a current time message.
3. If the current time message's time stamp is smaller than the log message's time stamp, subtract the former from the latter. If the current time message's time stamp is larger than the log message's time stamp, add 1024 to the latter and subtract the former from the result.
4. Add the resulting number to the current time message's *unix_time* to determine the log message's actual time.

Since log data is stored in a finite, circular buffer, some of the data can be overwritten before being read. If this happens, the following message appears at the appropriate place in the dump:

```
Log wrapped; data missing.
```

Note: If this message appears in the middle of a dump, which can happen under load, it indicates that not all of the log data is being written to the log. Increasing the size of the log with the **dfstrace setlog** command may alleviate this problem.

EXAMPLES

The following command dumps the log used by the **cm** kernel event set on the local machine:

```
# dfstrace dump cm
DFS Trace Dump -
   Date: Fri Oct  8 10:18:02 1993
Found 1 logs.
Contents of log cmfx:
Log wrapped; data missing.
time 520.211319, pid 25135: found a princ 62b4144 ref 3
time 520.211355, pid 25135: find a princ (fast path) 62b4144, ref 3
time 520.211387, pid 25135: fshs_GetPrincipal END 62b4144, ref 3
time 520.211411, pid 25135: fshs_PutPrincipal 62b4144 ref 3
time 520.219153, pid 25135: Lookup 8005a4d.81c6c35.0.3fe/param.h, flags 0x1
time 520.219440, pid 25135: fshs_GetPrincipal START
time 520.219483, pid 25135: fshs_GetHost, cookie 667de00
time 520.219511, pid 25135: fshs_FindHost, cookie 667de00
time 520.219559, pid 25135: find a prime host 62a2068
time 520.219590, pid 25135: find a host in fast path 62a2068
time 520.219625, pid 25135: fshs_FindPrincipal..
time 715.203951, pid 0: Current time: Mon Sep 20 13:05:15 1993
time 717.969835, pid 24621: fshs_GetPrincipal START
time 717.969881, pid 24621: fshs_GetHost, cookie 66eed80
time 718.969910, pid 24621: fshs_FindHost, cookie 66eed80
time 718.969959, pid 24621: find a prime host 62a2068
```

RELATED INFORMATION

Commands: **dfstrace lslog(8dfs)**, **dfstrace lsset(8dfs)**, **dfstrace setlog(8dfs)**.

dfstrace help

Purpose

Shows syntax of specified **dfstrace** commands or lists functional descriptions of all **dfstrace** commands

Synopsis

dfstrace help [**-topic** *string*][**-help**]

OPTIONS

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **setset**, not **dfstrace setset**). If this option is omitted, the output provides a short description of all **dfstrace** commands.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace help** command displays the first line (name and short description) of the online help entry for every **dfstrace** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **dfstrace apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

OUTPUT

The online help entry for each **dfstrace** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with **Usage:**, lists the command options in the prescribed order.

EXAMPLES

The following command displays the online help entry for the **dfstrace setset** command:

```
$ dfstrace help setset
/bin/dfstrace setset: set state of event sets
Usage: /bin/dfstrace setset  [-set <set_name>...]
[{-active | -inactive | -dormant}]
[-cdsentry <server entry in CDS>] [-help]
```

RELATED INFORMATION

Commands: **dfstrace apropos(8dfs)**.

dfstrace lslog

Purpose

Lists information on server process or kernel trace logs

Synopsis

```
dfstrace lslog{-set set_name -log log_name}[-long ] [-cdsentry
 server_entry_in_CDS][-help ]
```

OPTIONS

-set *set_name*
: Specifies the name of each event set whose corresponding logs you want to list. Specify this option or the **-log** option; omit both to list all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option.

-log *log_name*
: Specifies the name of each log that you want to list. Specify this option or the **-set** option; omit both to list all kernel logs on the local machine or all server process logs for the server process specified with the **-cdsentry** option.

-long
: Directs the **dfstrace lslog** command to also provide information on the size of each log in 4-kilobyte units (kwords) and whether the log is physically allocated in the kernel.

-cdsentry *server_entry_in_CDS*
: Specifies the full DCE pathname (***/.:/hosts/*** *machine****/*** *process_name*) of a server process whose logs you want to list. Use the **-set** or **-log** option with this option to list specific server process logs; use this option without the **-set** or **-log** option to list all logs associated with the server process. Omit this option to list kernel logs.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace lslog** command lists the specified server process or kernel trace logs. To display size and allocation information for the logs, run the command with the **-long** option. If you want to list a kernel log, it must reside on the local machine. If you want to list a server process log, it can reside on any machine; however, you must use the **-cdsentry** option to specify the appropriate server process.

To list a specific log, identify the log with the **-set** or **-log** option. Use the **-cdsentry** option to list a specific server process log, omit that option to list a specific kernel log.

To list all kernel logs on a machine, run the **dfstrace lslog** command without the **-set** or **-log** option. To list all server process logs associated with a particular server process, run the command with the **-cdsentry** option, but without the **-set** or **-log** option.

Privilege Required

To list a kernel log, the issuer must be logged in as **root** on the local machine. To list a server process log, the issuer must be listed in the **admin** list associated with that process on the machine specified by the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

OUTPUT

When run without the **-long** option, the **dfstrace lslog** command lists the logs only. When run with the **-long** option, the command lists the logs, the size of each log in kwords, and the allocation state of each log. There are two allocation states:

- **allocated** − Space is allocated for the log in the kernel or server process memory. This indicates that one or more of the event sets that write to this log are either **active** or **inactive**.
- **unallocated** − Space is *not* allocated for the the log in the kernel or server process memory. This indicates that all of the event sets that write to this log are **dormant**.

A log can also be **persistent**; however, the persistence of a log cannot currently be determined using **dfstrace** commands. If a log is **persistent**, it cannot be cleared during a global log clearing (executed by issuing **dfstrace clear** without the **-set** or **-log** option). Of course, the log can still be cleared if it is otherwise named with the **dfstrace clear** command. The **persistent** attribute prevents accidental clearing of important logs. The attribute is assigned to a log when the kernel or server process is compiled and cannot be changed.

EXAMPLES

The following command lists all kernel logs on the local machine:

```
# dfstrace lsl
Available logs:
cmfx
DFS syslog
```

The following command lists all server process logs used by the **flserver** process on the machine **dewitt**; it also provides the size and the allocation status of each log:

```
$ dfstrace lsl -long -cdsentry
/.:/hosts/dewitt/flserver
Available logs:
ubikvote : 30 kwords (allocated)
common : 30 kwords (allocated)
```

RELATED INFORMATION

Commands: **dfstrace lsset(8dfs)**, **dfstrace setlog(8dfs)**.

dfstrace lsset

Purpose

Lists server process or kernel event sets and their states

Synopsis

dfstrace lsset[**-set** *set_name*][**-cdsentry** *server_entry_in_CDS*][**-help**]

OPTIONS

-set *set_name*
: Specifies the name of each event set you want to list. Omit this option to list all kernel event sets on the local machine or all server process event sets for the server process specified with the **-cdsentry** option.

-cdsentry *server_entry_in_CDS*
: Specifies the full DCE pathname (***/.:/hosts/*** *machine**/*** *process_name*) of a server process whose event sets you want to list. Use this option with the **-set** option to list a distinct group of server process event sets; use this option alone to list all event sets associated with the server process. Omit this option to list kernel event sets.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace lsset** command lists the specified server process or kernel event sets and their states. If you want to list a kernel event set, it must reside on the local machine. If you want to list a server process event set, it can reside on any machine; however, you must use the **-cdsentry** option to specify the appropriate server process.

To list a specific event set, identify the event set with the **-set** option. Use the **-cdsentry** option to list a specific server process event set; omit that option to list a specific kernel event set.

To list all kernel event sets on a machine, run the **dfstrace lsset** command without any options. To list all server process event sets associated with a particular server process, run the command with the **-cdsentry** option only.

Privilege Required

To list a kernel event set, the issuer must be logged in as **root** on the local machine. To list a server process event set, the issuer must be listed in the **admin** list associated with that process on the machine specified with the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

OUTPUT

The command lists each event set and its state. There are three event set states:

- **active** − Tracing is enabled for the event set.

- **inactive** − Tracing is temporarily disabled for the event set; however, the event set continues to claim space occupied by the logs to which it sends data.
- **dormant** − Tracing is disabled for the event set; furthermore, the event set releases its claim to space occupied by the logs to which it sends data. When all of the event sets that send data to a particular log are in this state, the space allocated for that log is freed.

An event set can also be **persistent**. If an event set is **persistent**, its state cannot be set during a global state setting (executed by issuing **dfstrace setset** command with the **-set** option). Of course, the event set's state can still be set if the event set is otherwise specified with the **dfstrace setset** command. The **persistent** attribute prevents accidental resetting of an event set's state. The attribute is assigned to an event set when the kernel or server process is compiled and cannot be changed.

EXAMPLES

The following command lists all kernel event sets and their states on the local machine:

```
# dfstrace lss


Available sets:
cm: dormant
fx: dormant
zlc: dormant
fshost: active
xops: active
tkc: dormant
tkm/conflictQ: dormant
tkm/grants: dormant
tkm: dormant
tkm/revokes: dormant
episode/anode: dormant
episode/logbuf: dormant
episode/vnops: dormant
krpc: dormant
tkc: inactive
tpq: dormant
dfs_log: active persistent
```

The following command lists state information on the event set **ubikvote** for the **flserver** process on the machine **dewitt**:

```
$ dfstrace lss -set ubikvote -cdsentry /.:/hosts/dewitt/flserver
ubikvote: active
```

RELATED INFORMATION

Commands: **dfstrace clear(8dfs)**, **dfstrace setset(8dfs)**.

dfstrace setlog

Purpose

Sets the size of the indicated log

Synopsis

dfstrace setlog-log *log_name***-buffersize** *4-kilobyte_units* [**-cdsentry** *server_entry_in_CDS*][**-help**]

OPTIONS

-log *log_name*
Specifies the name of the kernel or server process log whose size you want to set.

-buffersize *4_kilobyte_units*
Defines the size of the log in 4-kilobyte units (kwords).

-cdsentry *server_entry_in_CDS*
Specifies the full DCE pathname (***/.:/hosts/*** *machine**/*** *process_name*) of a server process whose log size you want to set. Omit this option to set the size of a kernel log.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace setlog** command sets the size of a server process or kernel log. The size of the log is set in kwords. To set the size of a server process log, specify the server process with the **-cdsentry** option; to set the size of a kernel log, omit the **-cdsentry** option.

If a specified log is already allocated, it is cleared and freed when this command is run, and a new log of the desired size is created. Otherwise, a log of the desired size is created when the log is allocated.

To display the current size and allocated status of a log, issue the **dfstrace lslog** command.

Privilege Required

To set the size of a kernel log, the issuer must be logged in as **root** on the local machine. To set the size of a server process log, the issuer must be listed in the **admin** list associated with that process on the machine specified by the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

EXAMPLES

The following command sets the size of the **cmfx** kernel log to 64 kilobytes (16 kwords):

```
# dfstrace setl cmfx 16
```

The following command sets the size of the **ubikvote** server process log on the machine **dewitt** to 120 kilobytes (30 kwords):

```
$ dfstrace setl ubikvote 30 -cdsentry /.:/hosts/dewitt/flserver
```

RELATED INFORMATION

Commands: **dfstrace lslog(8dfs)**.

dfstrace setset

Purpose

Sets the state of server process or kernel event sets

Synopsis

dfstrace setset[**-set** *set_name*]{**-active -inactive -dormant** } [**-cdsentry** *server_entry_in_CDS*][**-help**]

OPTIONS

-set *set_name*
: Specifies the name of each event set whose state you want to set. Omit this option to set the state for all nonpersistent kernel event sets on the local machine or all nonpersistent server process event sets for the server process specified with the **-cdsentry** option.

-active
: Sets the state of each specified event set to **active**. Use this option or the **-inactive** or **-dormant** option.

-inactive
: Sets the state of each specified event set to **inactive**. Use this option or the **-active** or **-dormant** option.

-dormant
: Sets the state of each specified event set to **dormant**. Use this option or the **-active** or **-inactive** option.

-cdsentry *server_entry_in_CDS*
: Specifies the full DCE pathname (***/.:/hosts/*** *machine**/*** *process_name*) of a server process whose event set states you want to set. If this option is used with the **-set** option, only the states of the specified event sets are set; if this option is used without the **-set** option, the state of all nonpersistent event sets associated with the specified server process are set. Omit this option to set the state of kernel event sets.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **dfstrace setset** command sets the state of the server process or kernel event sets. To set the state of a kernel event set, you must run this command on the machine that contains that event set. To set the state of a server process event set, you can run the command from any machine; however, you must identify the corresponding server process by specifying the process with the **-cdsentry** option.

To set the state of a specific event set, identify the event set with the **-set** option. Use the **-cdsentry** option to set the state of a specific server process event set; omit that option to set the state of a specific kernel event set.

To set the state of each kernel event set on a machine, run the **dfstrace setset** command without the **-set** option. To set the state of each event set associated with a particular server process, run the command with the **-cdsentry** option, but without the **-set** option. Note that you cannot set the state of **persistent** event sets in this

global manner. The **persistent** attribute prevents accidental resetting of an event set's state. The attribute is assigned to an event set when the kernel or server process is compiled and cannot be changed.

The state of each event set is defined by using the **-active**, **-inactive**, or **-dormant** option. These options correspond to the following event set states:

- **active** − Tracing is enabled for the event set.
- **inactive** − Tracing is temporarily disabled for the event set; however, the event set continues to claim space occupied by the logs to which it sends data.
- **dormant** − Tracing is disabled for the event set; furthermore, the event set releases its claim to space occupied by the logs to which it sends data. When all of the event sets that send data to a particular log are in this state, the space for that log is deallocated.

Privilege Required

To set the state of a kernel event set, the issuer must be logged in as **root** on the local machine. To set the state of a server process event set, the issuer must be listed in the **admin** list associated with that process on the machine specified by the **-cdsentry** option (for example, **admin.fl** for the **flserver** process and **admin.ft** for the **ftserver** process).

EXAMPLES

The following command sets the event state of all kernel event sets on the local machine to **inactive**:

```
# dfstrace sets -inactive
```

The following command sets the event state of the event set **ubikvote** to **active** for the **flserver** process on the machine **dewitt**:

```
$ dfstrace sets -set ubikvote -active -cdsentry/.:/hosts/dewitt/flserver
```

RELATED INFORMATION

Commands: **dfstrace lsset(8dfs)**, **dfstrace setlog(8dfs)**.

flserver

Purpose

Initializes the Fileset Location (FL) Server

Synopsis

flserver[**-adminlist** *filename*][**-verbose**][**-help**]

OPTIONS

-adminlist *filename*
: Specifies the administrative list file that contains principals and groups authorized to execute **flserver** RPCs (usually using **fts** commands). If this option is omitted, the **flserver** process uses the default administrative list file, *dceloca*l**/var/dfs/admin.fl**.

-verbose
: Directs the **flserver** process to provide a detailed report on what it is doing by displaying messages on standard error.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **flserver** command. See the **bos help** and **bos apropos** reference pages for examples of using these commands.

DESCRIPTION

The Fileset Location (FL) Server maintains the Fileset Location Database (FLDB), which tracks the location of all DCE LFS and non-LFS filesets. The FL Server, or **flserver** process, must run on all Fileset Database machines. It is usually started and controlled by the BOS Server; if it is not, execute the **flserver** process as a background process. The binary file for the **flserver** process resides in *dceloca*l**/bin/flserver**.

The first time it is initialized, the **flserver** process creates the FLDB files in *dceloca*l**/var/dfs**; all FLDB files have a root name of **fldb**. The **flserver** process also creates the *dceloca*l**/var/dfs/admin.fl** administrative list file if the file does not already exist.

The principals and members of groups in the **admin.fl** administrative list are authorized to issue commands to create server entries and fileset entries in the FLDB. The list must also include the abbreviated DFS server principals of all Fileset Database machines to allow for the proper distribution of changes via the Ubik database synchronization mechanism. Because the FLDB is a replicated database, the **admin.fl** administrative lists for all **flserver** processes in a cell must contain the same principals and groups.

In addition, when the **flserver** process is first initialized, it makes a **ubik_ServerInit** call to register its existence as a server process with the Ubik coordinator. It then listens for incoming RPCs to which to respond.

Each time it is started, the **flserver** process creates the event log file *dceloca*l**/var/dfs/adm/FlLog** if the file does not already exist. It then appends

messages to the file. If the file exists when the **flserver** process is started, the process moves it to the **FlLog.old** file in the same directory (overwriting the current **FlLog.old** file if it exists) before creating a new version to which to append messages.

Privilege Required

The issuer must be logged in as **root** on the local machine.

OUTPUT

If problems are encountered during initialization, the **flserver** process displays error messages on standard error output. The **flserver** process keeps an event log in the file *dcelocal***/var/dfs/adm/FlLog**.

If run with the **-verbose** option, the **flserver** process provides a detailed report on what it is doing by displaying messages on standard error.

RELATED INFORMATION

Files: **admin.fl(4dfs)**, **FlLog(4dfs)**.

fms

Purpose

Determines tape size and End Of File (EOF) mark size for a tape drive

Synopsis

fms-device *device_name* [**-help**]

OPTIONS

-device *device_name*
Names the device name of the tape drive whose tape size and EOF mark size are to be reported. The format of this name varies with each operating system.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **fms** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The **fms** command is used with the Backup System to determine the tape size and EOF mark size for the tape drive indicated with **-device**. It is primarily useful for determining information required for specifying a tape drive's parameters in the **TapeConfig** file. If your drive supports compression, turn off compression prior to running this command.

Before issuing the command, insert a tape into the drive. Use a blank tape, one that can be recycled, or one to be initialized with file marks. The tape is overwritten while the command executes. Because this command inserts file marks onto the entire tape, it can take from several hours to more than a day to complete.

The command sends output to both the terminal and an **FMSLog** file that it creates in the directory it is to be issued from. The output reports the tape size and EOF mark size for the tape drive. It is recommended that the tape size returned by the command be reduced by 10 to 15% before being used in the **TapeConfig** file. It is also recommended that the EOF mark size be increased by 10 to 15% before being used in the **TapeConfig** file.

Privilege Required

Each time it is run, the **fms** command creates the **FMSLog** file if it does not already exist in the directory the command is issued from. In this case, the issuer of the command must have write, execute, and insert permissions on the current working directory. If the file already exists, the command truncates the file (clears its contents) before writing to it, in which case the issuer needs only write permission on the file.

OUTPUT

The **fms** command produces terminal output and an **FMSLog** file in the directory from which it is issued. The terminal output and **FMSLog** file list the tape capacity and the size of the EOF mark for the tape drive specified by **-device**.

The first few lines of output displayed on the screen and written to the **FMSLog** file include status information about the execution of the command, including such information as the number of blocks and the number of file marks written to the tape by the command. The last two lines of terminal and file output provide the following information:

Tape capacity is *number* **bytes**
Specifies the tape size, in bytes, for the tape drive.

File marks are *number* **bytes**
Specifies the file mark size, in bytes, for the tape drive.

If a problem with the tape drive prevents execution of the command, no **FMSLog** file is created and the message **Can't open tape device** *device_name* is displayed. If a problem prevents creation of the **FMSLog** file, the message **Can't open log file** is displayed. In both cases, execution of the command stops when the message is displayed.

EXAMPLES

The following command determines the EOF mark size for the tape drive whose device name is **/dev/rmt1h**:

```
$ fms /dev/rmt1h
```

The command displays the following output on the screen:

```
wrote block: 130408
Finished data capacity test - rewinding
wrote 1109 blocks, 1109 file marks
Finished file mark test
Tape capacity is 2136604672 bytes
File marks are 1910220 bytes
```

It writes the following information to the **FMSLog** file:

```
fms test started
wrote 130408 blocks
Tape capacity is 2136604672 bytes
File marks are 1910220 bytes
```

The tape drive used in the example uses tapes 2,136,604,672 bytes in size, and creates EOF marks of 1,910,220 bytes in size.

RELATED INFORMATION

Files: **FMSLog(4dfs)**, **TapeConfig(4dfs)**.

fts

Purpose

Introduction to the **fts** command suite

OPTIONS

The following options are used with many **fts** commands. They are also listed with the commands that use them.

-fileset { *name* **|** *ID***}**
Specifies the fileset to use with the command. You can specify either a fileset name or a fileset ID.

-server *machine*
Specifies the File Server machine to use with the command. This option is typically used to provide the name of the File Server machine on which the fileset or filesets to use with the command reside. You can use any of the following to specify the File Server machine:

1. The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*)
2. The machine's host name (for example, **fs1.abc.com** or **fs1**)
3. The machine's IP address (for example, **11.22.33.44**)

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition to use with the command. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the */dcelocal***/var/dfs/dfstab** file.

-cell *cellname*
Specifies that the command is to be run with respect to the cell named by the *cellname* argument. By default, commands are executed in the local cell of the issuer of the command.

-noauth
Directs the **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. Generally, the **-noauth** option is included with a command if DFS authorization checking is disabled on a server machine on which administrative privilege is required or if the Security Service is unavailable.

If DFS authorization checking is disabled, DFS processes require no administrative privilege to issue any command; any user, even the identity **nobody**, has sufficient privilege to perform any operation. If the Security Service is unavailable, a user's security credentials cannot be obtained.

DFS authorization checking is disabled with the **bos setauth** command or by including the **-noauth** option when the **bosserver** process is started on a machine. DFS authorization checking is typically disabled

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file

Include the **-noauth** option with a command that requires administrative privilege only if DFS authorization checking is disabled on the necessary machines. A command that requires administrative privilege fails if the **-noauth** option is included and DFS authorization checking is not disabled. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, */.../abc.com/hosts/fs1/dfs-server*. (Do not confuse a machine's DFS server principal with its unique **self** identity.)

Use this option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs the **fts** program to provide detailed information about its actions as it executes the command. This is useful mainly for debugging or trace purposes. The amount of additional information displayed when the -verbose option is specified varies for different commands.

-help Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see the **dfs_intro(8dfs)** reference page.

DESCRIPTION

Most commands in the **fts** command suite are administrative-level commands used only by system administrators to contact the Fileset Server and the Fileset Location Server (FL Server). (The primary exception is the **fts lsquota** command, which is also issued by users to determine the quota of filesets with which they work.) Commands in the **fts** suite are used to instruct the Fileset Server to create and delete filesets, as well as to move, replicate, and back up filesets. The FL Server automatically records in the Fileset Location Database (FLDB) any changes in fileset status and fileset location resulting from **fts** commands.

If the execution of an **fts** command is interrupted by a server or a process failure, subsequent execution of the command continues at the interruption point rather than at the beginning of the operation. Therefore, before executing a command, the Fileset Server and the FL Server verify that running the command has an effect. If the desired end-state already exists, the command is not executed; if the end-state does not yet exist, the command continues as necessary to achieve it.

If the issuer explicitly interrupts a fileset operation with an interrupt signal, the fileset is locked. The issuer must unlock it with the **fts unlock** command before proceeding.

DCE Local File System

The DCE Local File System (DCE LFS) is a high-performance, log-based file system. It supports the use of DCE LFS *aggregates*, which are physically equivalent to standard UNIX disk partitions but also contain a specialized log of *metadata* about the structure and location of information they house.

DCE LFS aggregates, in turn, support the use of DCE LFS filesets. DCE LFS filesets are hierarchical groupings of files managed as a single unit. They can vary in size but are almost always smaller than a disk partition. As a result, multiple DCE LFS filesets can be stored on a single aggregate. Non-LFS filesets occupy the entire partition on which they reside.

Because of the differences between DCE LFS and non-LFS filesets, the following **fts** commands function only with DCE LFS filesets. Refer to the appropriate command reference pages for more information about the functionality provided by these commands.

- **fts addsite(8dfs)**
- **fts clone(8dfs)**
- **fts clonesys(8dfs)**
- **fts create(8dfs)**
- **fts delete(8dfs)**
- **fts lsreplicas(8dfs)**
- **fts move(8dfs)**
- **fts release(8dfs)**
- **fts rmsite(8dfs)**
- **fts setquota(8dfs)**
- **fts setrepinfo(8dfs)**
- **fts statrepserver(8dfs)**
- **fts update(8dfs)**
- **fts zap(8dfs)**

Fileset Location Database Information

The Fileset Location Database (FLDB) is maintained by the Fileset Location Server (FL Server). A master copy of the FLDB is stored on one Fileset Database machine, with copies synchronized on other Fileset Database machines via the Ubik library of facilities. It is essential that the information in the FLDB correspond to the status of the filesets on the File Server machines. Therefore, any **fts** command that affects fileset status also changes the corresponding FLDB entry automatically. If an **fts** operation is interrupted before completion, information in the FLDB can differ from information on a File Server machine. In these cases, the **fts syncfldb** and **fts syncserv** commands must be used to align the information.

There is an entry in the FLDB for each read/write DCE LFS and non-LFS fileset. Each entry for a DCE LFS fileset also records information about the read-only and backup versions of the fileset because these versions do not have their own entries. The information in an FLDB entry includes the fileset's name and fileset ID number, the ID numbers of its read-only and backup versions (if it is a DCE LFS fileset), site definitions, site counts, and status flags. Complete details about the FLDB are included in Part 1 of this manual.

Fileset Header Information

A separate fileset header is stored at the site of each copy of a DCE LFS fileset, regardless of its type (read/write, read-only, or backup). The data structure of the fileset header records the physical memory addresses of the files in the fileset on the partition on which the fileset is stored. The fileset header binds all the files into a logical unit without requiring that they be stored in contiguous disk blocks.

The header of a DCE LFS fileset includes the following information: the fileset's name; its fileset ID number; its type (read/write, read-only, or backup); its size; the ID numbers of its parent, clone, and backup versions; its creation date; and the date of its last modification.

Cautions

Specific cautionary information is included with individual commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ man fts
Displays the reference page for the command suite.

$ man fts_ *command*
Displays the reference page for an individual command. You must use an _ (underscore) to connect the command suite to the command name. *Do not use the underscore when issuing the command in DFS.*

$ fts help
Displays a list of commands in a command suite.

$ fts help *command*
Displays the syntax for a single command.

$ fts apropos -topic *string*
Displays a short description of any commands that match the specified *string*.

Consult the **dfs_intro(8dfs)** reference page for complete information about the DFS help facilities.

Privilege Required

Most **fts** commands can be issued by users included in either the **admin.ft** file or the **admin.fl** file. Some commands require that the issuer be included on both lists; some commands also require that the user have certain permissions for a file or directory. Specific privilege information is listed with each command's description.

RELATED INFORMATION

Commands: **dfs_intro(8dfs)**, **fts addsite(8dfs)**, **fts aggrinfo(8dfs)**, **fts apropos(8dfs)**, **fts clone(8dfs)**, **fts clonesys(8dfs)**, **fts create(8dfs)**, **fts crfldbentry(8dfs)**, **fts crmount(8dfs)**, **fts crserverentry(8dfs)**, **fts delete(8dfs)**, **fts delfldbentry(8dfs)**, **fts delmount(8dfs)**, **fts delserverentry(8dfs)**, **fts dump(8dfs)**, **fts edserverentry(8dfs)**, **fts help(8dfs)**, **fts lock(8dfs)**, **fts lsaggr(8dfs)**, **fts lsfldb(8dfs)**, **fts lsft(8dfs)**, **fts lsheader(8dfs)**, **fts lsmount(8dfs)**, **fts lsquota(8dfs)**, **fts lsreplicas(8dfs)**, **fts lsserverentry(8dfs)**, **fts move(8dfs)**, **fts release(8dfs)**, **fts rename(8dfs)**, **fts restore(8dfs)**, **fts rmsite(8dfs)**, **fts setquota(8dfs)**, **fts setrepinfo(8dfs)**, **fts statftserver(8dfs)**, **fts statrepserver(8dfs)**, **fts syncfldb(8dfs)**, **fts syncserv(8dfs)**, **fts unlock(8dfs)**, **fts unlockfldb(8dfs)**, **fts update(8dfs)**, **fts zap(8dfs)**.

Files: **admin.fl(4dfs)**, **admin.ft(4dfs)**, **dfstab(4dfs).**

fts addsite

Purpose

Adds a replication site for a read/write DCE LFS fileset

Synopsis

```
fts addsite-fileset { name ID}-server  machine-aggregate name
[-maxsiteage  interval][-cell  cellname][{-noauth | -localauth }][-verbose ]
[-help]
```

OPTIONS

-fileset { *name* | *ID*}
Specifies the complete name or fileset ID number of the read/write source fileset for which the replication site is to be added.

-server *machine*
Names the File Server machine on which the replica is to be stored. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate on which the replica is to be stored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file.

-maxsiteage *interval*
Specifies the maximum amount of time that the replica to be stored at the site can be out-of-date (MaxSiteAge). The Replication Server attempts to keep the replica current within this amount of time. If this option is omitted, the DefaultSiteAge for the read/write site is used as the value for the MaxSiteAge. This option must be specified if the DefaultSiteAge was not defined for the read/write fileset (if the **-defaultsiteage** option was omitted when the **fts setrepinfo** was used to set the replication parameters for the read/write fileset).

Use this option only with Scheduled Replication. The MaxSiteAge of a replication site is ignored if Release Replication is used.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts addsite** command defines a replication site for the read/write DCE LFS fileset specified with the **-fileset** option. A replication site is a File Server machine and aggregate where a read-only replica of a read/write fileset is to be stored. The command also increments the number of fileset entries recorded as residing on the File Server machine specified with the -server option in the Fileset Location Database (FLDB) entry for the server.

A fileset's replication sites are recorded in the FLDB entry for the fileset. If this is the first replication site defined for the fileset, the status flag for the read-only version of the fileset is changed to **valid** in anticipation of the creation of a read-only fileset at the site.

Enter this command once for each replication site to be defined for a read/write fileset. Before this command is issued, the **fts setrepinfo** command must be used to set the replication parameters for the read/write fileset and a server entry must exist in the FLDB for the File Server machine specified with the **-server** option.

If Release Replication is used with the fileset, the first site defined with this command must be on the same File Server machine as the read/write, source fileset. If it is on the same aggregate as the source fileset, it is created as a clone of the source. Because it is created as a clone fileset, which has the same structure as a backup fileset, it shares data with the read/write fileset; therefore, it requires potentially much less space than a full read-only fileset created on a different aggregate.

A File Server machine can house only a single read-only version of a given read/write fileset. The command fails if an attempt is made to define a second replication site for a given fileset on the same File Server machine. Also, the File Server machine that houses a replication site must reside in the same cell as the read/write fileset for which the replication site is defined.

The FLDB entry for a read/write fileset records the locations of the fileset's replication sites; the server machine on which a site is defined must have a server entry in the FLDB that records the entry for the read/write fileset.

The FLDB can record a maximum of 16 sites for all versions of a fileset combined, a site being a specific File Server machine and aggregate. The read/write version and backup version (if it exists) of a fileset share a single site definition. If you define a replication site for a fileset at the same site as its read/write and backup versions, you can then define 15 more replication sites for the fileset; this approach allows you to define up to 16 replication sites. If you do not place a replica of a fileset at the same site as its read/write and backup versions, you can define a maximum of 15 replication sites for the fileset.

The **-maxsiteage** option is used to define the MaxSiteAge for the site, which is the maximum amount of time the replica at the site can be out-of-date. The Replication Server always attempts to copy the latest version of the fileset to the site before the

MaxSiteAge expires. Use the **-maxsiteage** option only if Scheduled Replication is used with the source fileset; the MaxSiteAge is ignored if Release Replication is used.

The DefaultSiteAge associated with the read/write fileset is used by default if the **-maxsiteage** option is omitted. The **-maxsiteage** option is required with the **fts addsite** command if the **-defaultsiteage** option was omitted when the **fts setrepinfo** command was used to define the replication parameters for the read/write fileset.

If Release Replication is used, the **fts release** command must be used to place a read-only replica at the replication site defined on the same File Server machine as the source fileset. The Replication Servers at the fileset's other replication sites then copy the replica to the sites on their respective machines. If Scheduled Replication is used, the Replication Servers at the replication sites automatically copy the source fileset to their sites. Immediate updates to sites using either type of replication can be forced with the **fts update** command.

Use the **fts aggrinfo** command to check the available space on an aggregate before adding it as a replication site. (Use the **fts lsft** command to check the size of the read/write fileset.) Use the **fts rmsite** command to remove a replication site and to instruct the Replication Server to remove the replica stored at the site. Use the **fts lsreplicas** command to determine the status of the read-only replica at a site.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or must own the server entry for each machine on which a version of the source fileset specified with the **-fileset** option resides.

EXAMPLES

The following command defines a read-only site for the fileset **rs_aix41.bin**. The site is defined as the aggregate whose device name is **/dev/lv01** on the File Server machine named **fs3**.

```
$ fts addsite rs_aix41.bin /.../abc.com/hosts/fs3 /dev/lv01
```

RELATED INFORMATION

Commands: **fts lsreplicas(8dfs)**, **fts release(8dfs)**, **fts rmsite(8dfs)**, **fts setrepinfo(8dfs)**, **fts update(8dfs)**.

Files: **dfstab(4dfs)**.

fts aggrinfo

Purpose

Displays disk space information about aggregates and partitions exported from a File Server machine

Synopsis

```
fts aggrinfo -server machine[-aggregate  name][-cell  cellname]
[-noauth | -localauth] [-verbose ][-help ]
```

OPTIONS

-server *machine*
: Names the File Server machine about whose aggregates and partitions information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of an exported aggregate or partition about which information is to be displayed. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file. If this option is omitted, information is provided about all of the exported aggregates and partitions on the specified machine.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts aggrinfo** command lists information about the total amount of disk space and the amount of disk space currently available on exported aggregates and partitions. The **-server** option is used to specify the File Server machine on which the aggregates and partitions reside. The **-aggregate** option can be used to specify

a single aggregate or partition about which information is to be displayed. If this option is omitted, information about all aggregates and partitions exported from the specified server is displayed. (Much of the information displayed by the **fts aggrinfo** command is specified in the *dcelocal***/var/dfs/dfstab** file.)

The **fts aggrinfo** command displays roughly the same information as the **df** command available in the UNIX operating system. The **df** command can also be used to display information about exported DCE LFS aggregates and locally mounted DCE LFS filesets.

The **fts lsaggr** command can also be used to list all of the aggregates and partitions exported from a File Server machine.

Privilege Required

No privileges are required.

OUTPUT

The **fts aggrinfo** command displays a separate line for each aggregate or partition. Each line displays the following information:

- The file system type (**LFS** for a DCE LFS aggregate, or **Non-LFS** for a non-LFS partition).
- The aggregate name.
- The device name.
- The number of kilobytes of disk space currently available for use on the aggregate or partition.
- The total number of kilobytes on the aggregate or partition (not including any reserved disk space).
- The number of kilobytes, if any, of reserved disk space on the aggregate or partition. DCE LFS reserves a variable amount of disk space on each aggregate for internal purposes (for example, to accommodate additional space required for fileset move and clone operations). Some non-LFS implementations also reserve some amount of overdraw disk space for administrative purposes.

The **fts aggrinfo** and UNIX **df** commands produce essentially the same information.

EXAMPLES

The following example displays information about the disk space available on all aggregates and partitions exported from the File Server machine named */.../abc.com/hosts/fs1*:

```
$ fts aggrinfom /.../abc.com/hosts/fs1
Non-LFS aggregate /usr (/dev/lv02): 24048 K free out of total 98304
     (10923 reserved)
Non-LFS aggregate /tmp (/dev/lv03): 11668 K free out of total 12288
     (1365 reserved)
LFS aggregate lfs1 (/dev/lfs1): 100537 K free out of total 101340
     (2048 reserved)
LFS aggregate lfs2 (/dev/lfs2): 71957 K free out of total 73728
     (2048 reserved)
```

RELATED INFORMATION

Commands: **fts lsaggr(8dfs)**.

Files: **dfstab(4dfs)**.

fts apropos

Purpose

Shows each help entry containing a specified string

Synopsis

fts apropos-topic *string*[**-help**]

OPTIONS

-topic *string*
: Specifies the keyword string for which to search. If it is more than a single word, surround the string with ″ ″ (double quotes) or other delimiters. Type all strings for **fts** commands in all lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts apropos** command displays the first line of the online help entry for any **fts** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **fts help** command.

Privilege Required

No privileges are required.

OUTPUT

The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **fts** command where the string specified by **-topic** is part of the command name or the first line.

EXAMPLES

The following command lists all **fts** commands that have the word **mount** in their names or short descriptions:

```
$ fts apropos mount
crmount: make mount point
delmount: remove mount point
lsmount: list mount point
```

RELATED INFORMATION

Commands: **fts help(8dfs)**.

fts clone

Purpose

Creates a backup version of a read/write DCE LFS fileset

Synopsis

fts clone -fileset { *name ID*}[**-cell** *cellname*][**-noauth** | **-localauth**] [**-verbose**][**-help**]

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the read/write source fileset.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

This command creates a backup version, or clone, of the indicated read/write DCE LFS fileset. It names the new backup version by adding a **.backup** extension to the name of the read/write source fileset. It places the backup version at the same site (File Server machine and aggregate) as the read/write version. The **fts clone** command *cannot* back up non-LFS filesets.

If no backup version exists, the command changes the status flag for the backup version in the fileset's entry in the Fileset Location Database (FLDB) to **valid**. It also increments the number of fileset entries recorded as residing on the File Server machine in the FLDB entry for the server.

If a backup version already exists, the new clone replaces it. The status flag for the backup version remains **valid**, and the number of fileset entries recorded in the File Server machine's FLDB entry remains unchanged.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine on which the read/write copy of the fileset is stored. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset resides.

EXAMPLES

The following command creates a backup version of the fileset *user.smith*:

```
$ fts clone user.smith
```

RELATED INFORMATION

Commands: **fts clonesys(8dfs)**.

fts clonesys

Purpose

Creates backup versions of all indicated read/write DCE LFS filesets

Synopsis

fts clonesys[**-prefix** *string*][**-server** *machine*][**-aggregate** *name*][**-cell** *cellname*][**-noauth** | **-localauth**][**-verbose**][**-help**]

OPTIONS

-prefix *string*
Specifies a character string of any length. Every fileset with a name matching this string is cloned. Include field separators (such as dots) if appropriate. This option can be combined with **-server**, **-aggregate**, or both. Omit all three options to back up all filesets in the local cell.

-server *machine*
Specifies the File Server machine where the read/write source filesets are stored. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. This option can be combined with **-prefix**, **-aggregate**, or both. Omit all three options to back up all filesets in the local cell.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate on **-server** where the read/write source filesets are stored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file. This option can be combined with **-server**, **-prefix**, or both. Omit all three options to back up all filesets in the local cell. The **-server** option must be specified if this option is used.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts clonesys** command creates a backup version, or clone, of each indicated read/write DCE LFS fileset. It names each backup version by adding a **.backup** extension to the name of its read/write source fileset. It places each backup version at the same site (File Server machine and aggregate) as its read/write version. The **fts clonesys** command *cannot* back up non-LFS filesets.

If no backup version of a fileset exists, the command changes the status flag for the backup version in the fileset's entry in the Fileset Location Database (FLDB) to **valid**. It also increments the number of fileset entries recorded as residing on the File Server machine in the FLDB entry for the server.

If a backup version of a fileset already exists, the new clone replaces it. The status flag for the backup version remains **valid**, and the number of fileset entries recorded in the File Server machine's FLDB entry remains unchanged.

The **fts clonesys** command returns a **0** if all DCE LFS filesets were successfully backed up, regardless of whether backups of any non-LFS filesets were attempted. The command returns a **1** if one or more DCE LFS filesets could not be backed up or if only backups of non-LFS filesets were attempted.

By combining the **-prefix**, **-server**, and **-aggregate** options, you can create backup copies of different subsets of read/write filesets. To back up

- All filesets in the local cell, specify no options.
- All filesets in the local cell with a name beginning with the same character string (for example, **sys.** or **user.**), specify the string with the **-prefix** option.
- All filesets on a File Server machine, specify the machine's name with the **-server** option.
- Filesets on a specific aggregate on a File Server machine, specify both the **-server** and **-aggregate** options.
- Filesets with a certain prefix on a specific File Server machine, specify both the **-prefix** and **-server** options.
- Filesets with a certain prefix on a specific aggregate on a File Server machine, specify the **-prefix**, **-server**, and **-aggregate** options.

Use the **fts clone** command to back up a single read/write DCE LFS fileset.

A DCE LFS fileset that is mounted locally (as a file system on its file server machine) cannot be backed up. You must remove its local mount point before attempting to clone the fileset.

Note: When you move a replicated DCE FLS fileset from one aggregate to another, the order of the sites listed in the FLDB entry for the fileset is modified. If the **fts closesys** command is used to clone a fileset that has one its read-only replication sites listed as the first site in the fileset's FLDB entry, the cloning fails.

To avoid this, either use the **fts clone** command, or use the **fts rmsite** command to remove each of the read-only sites and then add them again using the **fts addsite** command. This results in a reordered FLDB that lists the read/write site first, and the **ftd clonesys** command succeeds.

Privilege Required

The issuer must be listed in the **admin.ft** files on all machines on which read/write versions of the filesets are stored. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of any fileset to be backed up resides.

OUTPUT

If the **fts clonesys** command fails to back up either one or more DCE LFS filesets or one or more non-LFS filesets, the command displays the following output:

```
Total FLDB entries that were successfully backed up:
x (y failed; z wrong aggr type)
```

The variable **x** indicates the number of DCE LFS filesets that were successfully backed up. The variable **y** indicates the number of DCE LFS filesets that could not be backed up. The variable **z** indicates the number of non-LFS filesets that the command attempted to back up, but could not because of the command's inability to back up non-LFS filesets.

EXAMPLES

The following example creates a backup version of each fileset in the local cell whose name begins with the prefix **user.**:

```
$ fts clonesys -prefix user.
```

RELATED INFORMATION

Commands: **fts clone(8dfs)**.

Files: **dfstab(4dfs)**.

fts create

Purpose

Creates a read/write DCE LFS fileset and associated FLDB entry

Synopsis

```
fts create -ftname name-server machine -aggregate  name[-cell  cellname]
[-noauth | -localauth ] [-verbose ][-help ]
```

OPTIONS

-ftname *name*

Specifies a name for the read/write fileset. The name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The. (dot)
- The - (dash)
- The _ (underscore)

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. It can be no longer than 102 characters. This length does not include the **.readonly** or **.backup** extension, which is added automatically when a read-only or backup version of the fileset is created. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup filesets, so you cannot specify a fileset name that ends with either of these extensions.

-server *machine*

Names the File Server machine on which to create the new read/write fileset. A server entry for the machine must already exist. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*

Specifies the device name, aggregate name, or aggregate ID of the aggregate where the read/write fileset is to be stored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file.

-cell *cellname*

Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth

Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth

Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged

into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts create** command creates a read/write DCE LFS fileset, names it as specified by **-ftname**, and places it at the site specified by -server and **-aggregate**. The FL Server creates an entry for the fileset in the Fileset Location Database (FLDB) and allocates the fileset a unique ID number, which is recorded in both the fileset's FLDB entry and its fileset header. It also sets the status flag recorded for the read/write site in the fileset's FLDB entry to **valid** and increments the number of fileset entries recorded as residing on the specified File Server machine in the FLDB entry for the server. A server entry must exist in the FLDB for the File Server machine before this command is issued.

The FL Server also allocates and stores in the entry for the fileset in the FLDB the fileset ID numbers for the read-only and backup versions of the fileset that can be created later. It does not create these types of filesets or place anything at a read-only or backup site, so the status flags for the read-only and backup versions are set to **invalid**.

If this command succeeds, the fileset is available for use. It must be mounted in the file system with the **fts crmount** command for its contents to be visible in the global namespace. The command creates an empty root directory in the fileset, which becomes visible when the fileset is mounted. It records null ACLs as the default for use by the directory. (Although, due to the interaction between ACLs and UNIX mode bits, the directory has a set of implicit initial ACLs granting permissions to different users and groups.)

When a cell is initially configured, the **fts create** command is used to create the cell's main read/write fileset, **root.dfs**. Although **root.dfs** can be a non-LFS fileset, it must be a DCE LFS fileset if functionality such as replication is to be available in the cell. See Part 1 of this manual for information about configuring the root fileset to support replication.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine specified by **-server**. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for the machine specified by **-server**.

EXAMPLES

The following command creates the read/write fileset *user.pat*. The fileset is created on the aggregate **/dev/lv01** on the File Server machine **fs4**.

```
$ fts create user.pat /.../abc.com/hosts/fs4/dev/lv01
```

RELATED INFORMATION

Commands: **fts crfldbentry(8dfs)**, **fts crmount(8dfs)**.

Files: **dfstab(4dfs)**.

fts crfldbentry

Purpose

Creates a fileset entry in the FLDB

Synopsis

```
fts crfldbentry -ftname  name -server machine -aggrid ID [-cell cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-ftname *name*

Specifies a name for the fileset. The name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The. (dot)
- The – (dash)
- The _ (underscore)

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. It can be no longer than 102 characters. (Fileset names are restricted to this limit to accommodate the **.readonly** and **.backup** extensions that DCE LFS filesets of those types receive. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup DCE LFS filesets, so you cannot specify a fileset name that ends with either of these extensions.)

-server *machine*

Names the File Server machine on which the fileset resides. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggrid *ID*

Specifies the aggregate ID number to be assigned to the aggregate or partition in the Fileset Location Database (FLDB). The number specified with this option must also be used as the aggregate ID in the fourth field of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file on the machine where the aggregate or partition resides. The ID number must be a positive integer.

If the command is being used to create an FLDB entry for a non-LFS fileset (its typical use), the specified number must not already be in use in the **dfstab** file on the specified **-server**.

-cell *cellname*

Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth

Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts crfldbentry** command is used to register a fileset in the FLDB. After the fileset is registered, its location can be tracked by the FL Server. The command is typically used to create FLDB entries for non-LFS filesets.

Use the **-ftname** option to specify a name for the fileset according to the guidelines presented with the description of the **-ftname** option. Use the **-server** option to indicate the File Server machine that houses the aggregate or partition on which the fileset resides. Use the **-aggrid** option to specify an aggregate ID number to be associated with the aggregate or partition in the FLDB. This same number must be used in the entry for the aggregate or partition in the **dfstab** file on **-server** ; choose a number that is not already in use in the machine's **dfstab** file.

The FL Server allocates a unique fileset ID number for the fileset. This number, along with ID numbers allocated for read-only and backup filesets, is returned by the command. When creating an entry for a non-LFS fileset, the ID number allocated for the read-write fileset must be specified in the fifth field of the entry in the **dfstab** file for the partition on which the fileset resides.

The FL Server also sets the status flag for the read-write version in the fileset's entry to **valid**. In addition, it increments the number of fileset entries recorded as residing on the specified File Server machine in the FLDB entry for the server. A server entry must exist in the FLDB for the File Server machine before this command is issued.

After issuing this command to register a non-LFS fileset, create an entry for the partition on which the fileset resides in the local **dfstab** file, export the partition with the **dfsexport** command, and mount the fileset with the **fts crmount** command to make the fileset accessible in the DCE namespace. The **fts crserverentry** command must be used before this command to create a server entry in the FLDB for the machine on which the fileset resides.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for the machine specified by **-server**.

EXAMPLES

The following example creates an entry in the FLDB for a non-LFS fileset named *user.jlw*. The fileset is located on the File Server machine named **fs3**. The aggregate ID of the partition the fileset resides on is **7**.

```
$ fts crfldbentry user.jlw /.../abc.com/hosts/fs37
```

RELATED INFORMATION

Commands: **dfsexport(8dfs)**, **fts create(8dfs)**, **fts crserverentry(8dfs)**, **fts crmount(8dfs)**.

Files: **dfstab(4dfs)**.

fts crmount

Purpose

Creates a mount point for a fileset

Synopsis

```
fts crmount -dir directory_name{-fileset { name ID}-global }[-rw ][-fast ]
[-help ]
```

OPTIONS

-dir *directory_name*
: Names the location in the file tree at which the root directory of the fileset is to be mounted. The specified location becomes the fileset's mount point. The specified mount point must not already exist, but the parent directory of the mount point must exist in the DFS filespace.

-fileset { *name* **|** *ID***}**
: Specifies the complete name or fileset ID number of the fileset to be mounted. The mount point for the fileset is created at the location specified with the **-dir** option. The read/write, read-only, or backup version of the fileset can be named. Use this option or use the **-global** option.

-global
: Specifies that the mount point named with the **-dir** option is for the root of the DCE global namespace (a global root mount point). Use this option or use the **-fileset** option. Do not use the **-global** option under normal circumstances; it is maintained to provide backward compatibility with other file systems (for example, AFS).

-rw
: Specifies the type of the mount point as read/write. By default, a mount point is regular. If this option is used, the **-fileset** option must specify the read/write version of the fileset.

  An important function of the **-rw** option is to mount a cell's main read/write fileset, **root.dfs**, below the top level of the cell's DFS filespace. The option must be used in this capacity at installation if replication is to be available in the cell. See the description section for more information about the different types of mount points and about using the **-rw** option to create a read/write mount point for the fileset **root.dfs**.

-fast
: Directs **fts** not to verify the existence of the fileset indicated with the **-fileset** option. Use this option to create a mount point for a fileset that does not yet exist.

  By default, **fts** contacts the Fileset Location (FL) Server for the cell that houses the parent directory of the name supplied with the **-dir** option to verify that the FLDB contains an entry for the fileset to be mounted. If no FLDB entry exists for the specified fileset, the command displays a warning. The specified mount point is always created, regardless of whether the **-fast** option is provided; the option simply suppresses the check and any possible warnings.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts crmount** command creates a mount point for the fileset specified with the **-fileset** option at the location in the file tree specified with the **-dir** option. The mount point makes the contents of the specified fileset visible and accessible to users. Once this command is used to mount a fileset, no further actions need to be taken to mount the fileset and the fileset never needs to be mounted again.

A mount point is actually a special symbolic link that acts as an association between a directory location and a fileset. Mount points look and function like standard directories. When the Cache Manager encounters a mount point in a pathname traversal, it determines which fileset is associated with the mount point. When it finds that fileset, it can access files or directories contained in the fileset's root directory. It traverses any paths leading through directories or other mount points in the fileset until it finds the directory or file indicated by the pathname.

To a large extent, the type of a mount point determines the version of a fileset through which the Cache Manager searches for a requested directory or file. By default, every mount point created by the **fts crmount** command is a regular mount point, which is the most common type of mount point. However, if you include the **-rw** option with the command, the command creates a read/write mount point.

When the Cache Manager encounters a regular mount point, it checks the version of the fileset that the mount point indicates. If the mount point indicates the read-only or backup version of the fileset, the Cache Manager accesses that version. If the mount point indicates the read/write version of the fileset, the Cache Manager considers the fileset in which the mount point resides and acts accordingly, as follows:

- If a regular mount point that names a read/write fileset resides in a read-only fileset, the Cache Manager first determines whether the fileset is replicated. If the fileset is not replicated, the Cache Manager attempts to access the read/write version of the fileset; if the read/write version does not exist or is inaccessible, the Cache Manager cannot access the fileset. If the fileset is replicated, the Cache Manager attempts to access a read-only version of the fileset; if all of the read-only copies are unavailable, the Cache Manager cannot access the fileset (it does not attempt to access the read/write version of the fileset).
- If a regular mount point that names a read/write fileset resides in a read/write fileset, the Cache Manager attempts to access only the read/write version of the fileset. If the read/write version does not exist or is inaccessible, the Cache Manager cannot access the fileset.

When the Cache Manager encounters a read/write mount point, it attempts to access only the read/write version of the fileset, regardless of the type of fileset in which the mount point resides. If the read/write version of the fileset does not exist or is inaccessible, the Cache Manager cannot access the fileset.

Regular mount points promote greater fileset availability than read/write mount points because they allow the Cache Manager to access read-only filesets as often as possible. Because multiple copies of read-only filesets typically exist, regular mount points generally increase fileset availability. Conversely, because only a single version of a read/write fileset can exist, read/write mount points generally reduce fileset availability.

You typically mount filesets with regular mount points. A regular mount point is explicitly not a ″read-only″ mount point. Although the Cache Manager attempts to

access a read-only version of a fileset when it encounters a regular mount point, it accesses the read/write version of the fileset if no read-only versions of the fileset exist or if it is traversing a read/write path (that is, if it accessed the mount point in a read/write fileset).

During a cell's configuration, an important function of the **fts crmount** command is to create a mount point for the cell's main read/write fileset, **root.dfs**. The command must be used with the **-rw** option to create an explicit read/write mount point for the fileset below the top level of the cell's DFS filespace. The mount point for the fileset must be created at **/.../** *cellname***/fs/.rw**.

The **root.dfs** fileset is the first fileset mounted in a cell. The Cache Manager automatically mounts it at the top level of the cell's DFS filespace (**/.../** *cellname***/fs** by default, but it can be defined as something else). It must be created as a DCE LFS fileset with the **fts create** command if functionality such as replication is to be available in the cell.

Once the **root.dfs** fileset is mounted with a read/write mount point, it can be replicated. Replication is then available for DCE LFS filesets in the cell. If **root.dfs** is replicated before its read/write mount point is created with the **fts crmount** command, the read/write version of **root.dfs** cannot be accessed in the cell. See Part 1 of this manual for information about configuring **root.dfs** to support replication.

Privilege Required

If the parent directory of the mount point (the directory in which the mount point is to be created) is in a DCE LFS fileset, the issuer must have write, execute, control, and insert permissions on the directory. If the parent directory is in a non-LFS fileset, the issuer must have write and execute permissions on the directory.

CAUTIONS

Do not mount a fileset at more than one location in the file system. Creating multiple mount points can distort the hierarchical nature of the file system. Because the Cache Manager stores only a single pointer to the parent directory of the mount point for each fileset, it can become confused about which pathname to follow when searching for a file in a fileset with multiple mount points. This is true even if the full pathname of a file is specified.

Create multiple mount points for a fileset sparingly, only in a very limited number of troubleshooting and testing situations. Remove the extraneous mount points as soon as they are no longer necessary.

Do not create a symbolic link that begins with a # (number sign) or a **%** (percent sign) character. Because a mount point is a special type of symbolic link that uses these characters internally to identify its type, the Cache Manager becomes confused if it encounters a normal symbolic link that begins with one of these characters.

EXAMPLES

The following command creates a regular mount point (the default type of mount point) for the read/write fileset named *user.jlw* at the directory named */.../abc.com/fs/usr/jlw*.

```
$ fts crmount /.../abc.com/fs/usr/jlw user.jlw
```

The following command creates a read/write mount point for the fileset named **root.dfs** in the cell *abc.com*. The fileset is mounted at **.rw**, immediately below the top level of the cell's DFS filespace.

```
$ fts crmount /.../abc.com/fs/.rw root.dfs -rw
```

RELATED INFORMATION

Commands: **dfsd(8dfs)**, **fts create(8dfs)**, **fts delmount(8dfs)**, **fts lsmount(8dfs)**.

fts crserverentry

Purpose

Creates a server entry in the FLDB

Synopsis

fts crserverentry -server *machine***-principal** *name*[**-quota** *entries*][**-owner** *group*] [**-cell** *cellname*][**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-server *machine*
Specifies the server machine for which an entry is to be created in the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-principal *name*
Specifies the abbreviation for the DFS server principal to be registered in the FLDB for the machine (for example, **hosts/** *hostname*). The machine's principal name in the Registry Database must match this name.

-quota *entries*
Sets a limit on the number of fileset entries (read-write, read-only, and backup) in the FLDB that can be associated with the specified **-server**. If this option is omitted, a value of **0** (zero) is used, meaning an unlimited number of fileset entries can be associated with the specified File Server machine.

-owner *group*
Specifies the name of the group that is the owner of the server entry. A group can be specified by a full or abbreviated group name (for example, **/.../** *cellname***/** *group_name* or just *group_name*). Foreign groups cannot own a local server entry. If this option is omitted, no group owns the server entry. (The value **<nil>** is reported as the owner.)

-cell *cellname*
Specifies the cell in whose FLDB the server entry is to be created. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts crserverentry** command creates a server entry in the FLDB for the server machine specified with **-server**. You must issue this command for a server machine before issuing any other **fts** commands involving that machine (for example, before creating filesets on the machine with the **fts create** command, before adding the machine as a replication site with the **fts addsite** command, before moving filesets to the machine with the **fts move** command, and so on).

The **-quota** option is used to limit the number of filesets (read/write, read-only, and backup) that can reside on the specified File Server machine. When a fileset entry in the FLDB is defined to reference the File Server machine as the site of a fileset version, the FL Server increments the number of fileset entries recorded as residing on the server in its server entry. The FL Server creates no more than the specified **-quota** of fileset entries on the server machine.

The following commands update the number of fileset entries recorded for a File Server machine in its server entry. The **fts create**, **fts crfldbentry**, **fts addsite**, **fts restore**, **fts clone**, and **fts clonesys** commands increment the number of fileset entries recorded for the server; the **fts delete**, **fts delfldbentry**, and **fts rmsite** commands decrement the number of fileset entries recorded; the **fts move** command decrements and increments the number of fileset entries recorded on the source and destination machines, respectively; and the **fts syncfldb** and **fts syncserv** commands can update the number of fileset entries recorded, as necessary.

The **-owner** option is used to specify a group of administrators who can administer entries in the FLDB for all of the filesets on the specified File Server machine. The same group can be given ownership of all server entries for the File Server machines in the domain where the specified machine resides. Members of the group can then manipulate the FLDB entries for all of the filesets in the domain where the File Server machine resides. This way, the administrators in the group need not be included on the **admin.fl** list for the entire cell, which would allow them to modify all of the fileset entries in the FLDB in that cell.

Use the **fts lsserverentry** command to display the current values from the FLDB for a server entry. Use the **fts edserverentry** command to change the current values in the FLDB for a server entry. Use the **fts delserverentry** command to remove a server entry from the FLDB.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines.

EXAMPLES

The following example adds a server entry to the FLDB for a server machine named **fs1**. The abbreviated DFS server principal of the machine is specified with the **-principal** option as **hosts/fs1**. Because they are omitted, the **-quota** and **-owner** options receive the default values of **0** (zero) and the empty group ID (displayed as **<nil>**), respectively.

```
$ fts crserverentry /.../abc.com/hosts/fs1 hosts/fs1
```

RELATED INFORMATION

Commands: **fts delserverentry(8dfs)**, **fts edserverentry(8dfs)**, **fts lsserverentry(8dfs)**.

fts delete

Purpose

Removes a specified read/write or backup version of a DCE LFS fileset

Synopsis

```
fts delete -fileset {name ID}-server  machine -aggregate name[-cell cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* | *ID*}
Specifies the complete name or fileset ID number of the read/write or backup fileset to be removed. Include the **.backup** extension if specifying the name of a backup fileset.

-server *machine*
Names the File Server machine from which to remove the fileset. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate from which to remove the fileset. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the *dcelocal***/var/dfs/dfstab** file.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts delete** command removes the read/write or backup DCE LFS fileset indicated by the **-fileset** option from the site specified by the **-server** and **-aggregate** options. Versions of the fileset are removed and the Fileset Location Database (FLDB) entry for the fileset updated to record the removals as follows:

- *Removing a read/write fileset* automatically removes its associated backup version (if the backup version exists). If the fileset is replicated, you must use the **fts rmsite** command to remove all replication sites and read-only replicas of the fileset before you issue the **fts delete** command. The **fts rmsite** command removes a replication site and instructs the Replication Server to remove the replica stored at the site; if Release Replication is used for the fileset, you must also remove the replication site and replica stored on the same File Server machine as the read/write fileset. The **fts delete** command removes the entire entry for the fileset from the FLDB.
- Removing a backup fileset removes site information for the backup version from the fileset's FLDB entry and marks the backup version as **invalid** but does not erase its fileset ID number. Read/write and read-only versions of the fileset are not affected.

The number of fileset entries recorded in the server entry in the FLDB for the File Server machine from which a read/write or backup version of a fileset is removed is decremented once for each deleted version of the fileset. (Note that, if the indicated version of a fileset does not exist at the specified site but is referenced in the fileset's FLDB entry, the command removes site information about that version of the fileset from the fileset's entry and generally performs all other operations as indicated.)

Before you remove the read/write (and backup) version of a fileset, use the **fts rmsite** command to remove the fileset's replication sites and to instruct the Replication Server to remove the replicas stored at the sites. If Release Replication was used for the fileset, use the **fts rmsite** command to remove the replication site and replica stored at the read/write fileset's site as well.

After removing a fileset, use the **fts delmount** command to remove its mount point. Note that it might be better in some cases to remove a fileset's mount point before deleting the fileset; removing the mount point first ensures that no new access to the fileset is obtained. Users already accessing the data are unaffected by mount point removal.

If the DCE LFS fileset to be removed is also mounted locally (as a file system on its File Server machine), you must remove its local mount point before you delete it; the **fts delete** command cannot be used to delete a fileset that is mounted locally. In addition, because the backup version of a fileset is removed when its read/write version is removed, you cannot remove the read/write version of a fileset if its backup version is mounted locally. You must remove the backup version's local mount point before deleting the read/write version.

The **fts delfldbentry** command can be used to remove an FLDB entry for a fileset. Use the command only when you are certain that a fileset deletion was not recorded in the FLDB. The **fts zap** command can be used to remove a fileset when it is urgent that the fileset be removed but the FLDB is inaccessible. When the FLDB is again accessible, use the **fts delfldbentry** command to remove the fileset's FLDB entry or use the **fts syncfldb** and **fts syncserv** commands to synchronize the FLDB with the state of the filesets.

The **fts delfldbentry** command is also used to remove the FLDB entry for a non-LFS fileset. The **fts delmount** command is then used to remove its mount point, and the **dfsexport** command is used to detach the partition it resides on from the global namespace.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine specified by **-server**. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be deleted resides.

Output

The **fts delete** command displays the following error message if you attempt to delete a read/write fiileset for which one or more replication sites exist:

```
There is a RO copy in existence - please 'rmsite' before 'delete'.
```

Use the **fts rmsite** command to remove all replication sites for the fileset and reissue the **fts delete** command.

EXAMPLES

The following command deletes the read/write fileset named *user.terry* and its backup version (if it exists) from the aggregate named **/dev/lv01** on the File Server machine named **fs3**:

```
$ fts delete user.terry /.../abc.com/hosts/fs3/dev/lv01
```

RELATED INFORMATION

Commands: **dfsexport(8dfs)**, **fts delfldbentry(8dfs)**, **fts delmount(8dfs)**, **fts rmsite(8dfs)**, **fts syncfldb(8dfs)**, **fts syncserv(8dfs)**, **fts zap(8dfs)**.

Files: **dfstab(4dfs)**.

fts delfldbentry

Purpose

Removes a specified entry from the FLDB

Synopsis

```
fts delfldbentry {-fileset { name ID}-prefix  string}[-server  machine]
[-aggregate name][-cell  cellname][-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of a fileset. The entire entry for the fileset is removed from the Fileset Location Database (FLDB), regardless of whether the read/write, read-only, or backup version of the fileset is specified. Provide this option or use the **-prefix** option.

-prefix *string*
Specifies a character string of any length. Every FLDB entry that lists a fileset whose name begins with this exact string is removed (unless more restrictive constraints are specified with the **-server** and optionally **-aggregate** options). Include field separators such as dots if appropriate. Provide this option (optionally with **-server** and **-aggregate**) or use the **-fileset** option.

-server *machine*
Names a File Server machine. If a fileset's name matches the specified -prefix and it is listed as residing on the specified File Server machine, its entry is removed from the FLDB. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. If this option is used, **-prefix** must be used; **-aggregate** can also be used.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of an aggregate or partition on **-server**. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file. If a fileset's name matches the specified **-prefix** and it resides on the specified aggregate on **-server**, its entry is removed from the FLDB. If this option is used, **-prefix** and -server must be used.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts delfldbentry** command removes the entries for all indicated filesets from the FLDB. Regardless of the version of a fileset (read/write, read-only, or backup) specified with the command, the fileset's entire entry is removed. The command has no effect on actual filesets on File Server machines, only on their FLDB entries.

The command also decrements the number of fileset entries recorded in server entries, as appropriate. For each version of a fileset whose entry is removed from the FLDB, the number of fileset entries recorded in the server entry for the File Server machine it resides on is reduced by one.

By using the **-fileset** option alone or combining the **-prefix**, **-server**, and **-aggregate** options in increasingly specific ways, FLDB entries can be removed for varying numbers of filesets. To remove the FLDB entry for

- A single fileset, specify **-fileset**.
- Every fileset whose name begins with a certain character string (for example, **sys.** or **user.**), regardless of site, specify **-prefix** .
- Every fileset whose name begins with a certain character string and that is stored on a specific File Server machine, specify **-prefix** and **-server**.
- Every fileset whose name begins with a certain character string and that is stored on a specific aggregate of a specific File Server machine, specify -prefix, **-server**, and **-aggregate**.

This command can be used if the issuer is certain that a fileset removal is not recorded in the FLDB and does not want to take the time to synchronize an entire File Server machine. It can also be used to remove the FLDB entry for a non-LFS fileset to be removed from the global namespace. (Use the **fts rmsite** command to remove an incorrect entry for a read-only site from the FLDB.)

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset whose FLDB entry is to be removed.

CAUTIONS

Do not use this command as the standard way to remove a fileset entry from the FLDB. It can make the FLDB inconsistent with the filesets on server machines. Use the **fts delete** command to remove the fileset entry from the FLDB at the same time that the fileset is deleted. Also, because it can be used to remove multiple FLDB entries simultaneously, use this command carefully.

EXAMPLES

The following command removes the FLDB entry for the fileset **user.temp**:

```
$ fts delfldbentry user.temp
```

The following command removes all FLDB entries for filesets whose names begin with **test** and that are stored on the File Server machine named **fs3**:

```
$ fts delfldbentry -prefix test -server /.../abc.com/hosts/fs3
```

RELATED INFORMATION

Commands: **fts clone(8dfs)**, **fts delete(8dfs)**, **fts rmsite(8dfs)**, **fts syncfldb(8dfs)**, **fts syncserv(8dfs)**, **fts zap(8dfs)**.

Files: **dfstab(4dfs)**.

fts delmount

Purpose

Removes a mount point

Synopsis

fts delmount -dir *directory_name*[**-help**]

OPTIONS

-dir *directory_name*
Names the mount point to be deleted. Provide a complete pathname. The last element in the pathname must be an actual name, not. (dot) or.. (dot dot).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts delmount** command removes the mount point specified by **-dir**. The associated fileset is not affected, but it is inaccessible if no other mount points exist for it. When the mount point for a fileset is removed, any fileset mounted only as a subdirectory of the fileset's root directory becomes inaccessible.

Removing the mount point ensures that no new access to the fileset is obtained. Users already accessing data are unaffected by mount point removal.

Privilege Required

If **-dir** resides in a directory in a DCE LFS fileset, the issuer must have write, execute, and delete permissions on the directory in which it resides. If **-dir** resides in a directory in a non-LFS fileset, the issuer must have write and execute permissions on the directory in which it resides.

EXAMPLES

The following command removes the mount point for the fileset *user.vijay*, which is mounted at */.../abc.com/fs/usr/vijay*:

```
$ fts delm /.../abc.com/fs/usr/vijay
```

RELATED INFORMATION

Commands: **fts crmount(8dfs)**, **fts lsmount(8dfs)**.

fts delserverentry

Purpose

Deletes a server entry from the FLDB

Synopsis

fts delserverentry -server *machine*[**-cell** *cellname*][**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-server *machine*
Specifies the server machine whose entry is to be removed from the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-cell *cellname*
Specifies the cell from whose FLDB the server entry is to be removed. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts delserverentry** command removes the existing server entry from the FLDB for the server machine specified with **-server**. When the command is issued, no fileset entries in the FLDB can reference the server entry to be removed as the site of a fileset. If a fileset entry in the FLDB references the server entry to be removed, the command fails.

Use the **fts crserverentry** command to create a server entry in the FLDB. Use the **fts lsserverentry** command to display the current values from the FLDB for a server entry. Use the **fts edserverentry** command to change the current values in the FLDB for a server entry.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines.

EXAMPLES

The following example deletes the server entry from the FLDB for the server machine named **fs1**. No filesets can reside on the machine when the command is issued.

```
$ fts delserverentry /.../abc.com/hosts/fs1
```

RELATED INFORMATION

Commands: **fts crserverentry(8dfs)**, **fts edserverentry(8dfs)**, **fts lsserverentry(8dfs)**.

fts dump

Purpose

Converts a fileset to a bytestream format and places it in a file

Synopsis

```
fts dump -fileset { name | ID}{-time { date | 0}-version  number}[-server  machine]
[-file  filename][-cell  cellname][-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the fileset to be dumped. The read/write, read-only, or backup version of the fileset can be dumped. Append the **.readonly** or **.backup** extension to the name of the fileset to dump the read-only or backup version instead of the read/write version.

-time { *date* **|** *0***}**
Specifies a full or incremental dump. Three values are legal:

0 (zero)
A **0** (zero) value causes a full dump of the current version of the fileset.

*mm***/***dd***/***yy*
A month/day/year value causes an incremental dump from 12:00 a.m. (00:00) on the indicated date (for example, **1/23/90** or **10/16/92**). Only files with modification time stamps equal to or later than the specified date and time are dumped.

*mm***/***dd***/***yy hh***:***mm*
An exact date and time value causes an incremental dump from the specified time on the indicated date. Only files with modification time stamps equal to or later than the specified date and time are dumped. The time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The date format is the same as for a date alone. Surround the entire argument with ″ ″ (double quotes) because it contains a space (for example, ″**1/23/90 22:30**″ or ″**10/16/92 3:45**″).

Use this option to perform a full dump or to perform an incremental dump of only those files in the fileset modified since a specific date or date and time. Use this option or use **-version**.

-version *number*
Specifies an incremental dump based on the indicated fileset version number. Each DCE LFS fileset has a version number. Each file in the fileset records the version number that was current when the file was last modified. If this option is specified, only those files with version numbers equal to or greater than the specified version number are dumped. (A DCE LFS fileset's version number is recorded in its fileset header; it has the same format as a fileset ID number. Use the **fts lsheader** or **fts lsft** command to display a fileset's current version number.)

Use this option or use **-time**. Use this option only with DCE LFS filesets.

-server *machine*
Names the File Server machine that houses the version of the fileset to be dumped. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

This option is useful for dumping a particular read-only replica of a DCE LFS fileset for which multiple replicas exist. If you include the **.readonly** extension with the name of a fileset specified with the **-fileset** option, or if you specify the ID number of the read-only version of a fileset with the **-fileset** option, you can use the **-server** option to indicate the machine that houses the specific replica to be dumped. If you omit the **-server** option in these cases, the command dumps the replica that resides at the fileset's oldest read-only site (the replica at the site that has been defined for the longest time).

This option is always unnecessary if the read/write or backup version of a fileset is to be dumped.

-file *filename*
Specifies the complete pathname of the file to which the dump is to be written. If a complete pathname is not specified, the file is written to the current working directory. If this option is omitted, the data is sent to standard output (**stdout**).

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts dump** command converts the contents of the indicated fileset to a bytestream format. It puts the converted contents into the file specified with the **-file** option. If this option is omitted, the dumped data is sent to **stdout**. Both non-LFS and read/write, read-only, and backup DCE LFS filesets can be dumped.

Note: On DCE 2.2 for AIX, dump of non-LFS filesets is not supported.

The command's options can be used to perform the following types of dumps:

- A value of 0 (zero) specified with the **-time** option causes a full dump of the fileset.
- A date specified with the **-time** option causes an incremental dump of all files modified since 12:00 a.m. (00:00) on that date.
- A date and time specified with the **-time** option cause an incremental dump of all files modified since that date and time.
- A version number specified with **-version** causes an incremental dump of all files in the fileset with version numbers equal to or greater than the specified version number.

Dumping a fileset does not affect its status in the Fileset Location Database (FLDB) or at the site from which it is dumped. However, it does make the fileset inaccessible for the duration of the dump operation. For this reason, it is customary to dump the backup version of a fileset to prevent the read/write version from being inaccessible for an extended time.

If a read-only replica of a DCE LFS fileset is to be dumped and multiple replicas of the fileset exist, the **-server** option can be used to name the File Server machine that houses the specific replica to be dumped. Indicating a specific replica can be useful if network or hardware problems caused only some of a fileset's replicas to be updated. It can be especially useful for restoring the read/write version of a fileset that was lost before all of its replicas were updated, since you can dump and restore a specific replica that was updated before the read/write version was lost. (By default, all replicas of the same fileset are always identical; to determine whether all replicas of a fileset are the same, use the **fts lsft** command to display information about specific replicas.)

Note: On DCE 2.2 for AIX, dump of non-LFS filesets is not supported.

The **fts restore** command can be used to restore a fileset dumped with the **fts dump** command. You can use the **fts restore** command to restore a dump file to any type of fileset (DCE LFS or non-LFS), regardless of the type of fileset from which it was created. Thus, you can dump and restore data between DCE LFS and non-LFS filesets, as well as between different types of non-LFS filesets. (See the documentation for the **fts restore** command for more information about dumping and restoring filesets between different types of file systems.)

You cannot restore a fileset dumped in one cell to a site in another cell.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine on which the fileset is stored. In addition, the issuer must have the write, execute, and insert permissions on the directory in which the dump file is to reside.

EXAMPLES

The following command executes a full dump of the fileset *user.terry* into the file named */tmp/terry.dump*:

```
$ fts dump user.terry -time 0 /tmp/terry.dump
```

The following command executes an incremental dump of the fileset *user.smith* into the file named */tmp/smith.013191.dump*. Only those files in the fileset with

modification time stamps equal to or later than 6:00 p.m. on 31 January 1991 are included in the dump.

```
$ fts dump user.smith -time "1/31/91 18:00" /tmp/smith.013191.dump
```

RELATED INFORMATION

Commands: **fts lsft(8dfs)**, **fts restore(8dfs)**.

fts edserverentry

Purpose

Edits a server entry in the FLDB

Synopsis

```
fts edserverentry -server machine [{-rmaddr | -addaddr address | -changeaddr
 address}][-principal  name][-quota  entries][{-owner group | -noowner }]
[-cell  cellname][-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-server *machine*
: Specifies the server machine whose entry in the Fileset Location Database (FLDB) is to be edited. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. If the -rmaddr, **-addaddr**, or **-changeaddr** option is used with the command, specify the network address.

-rmaddr
: Removes the address specified with **-server** from the server entry identified by **-server** in the FLDB. If the name of the machine rather than one of its addresses is specified with **-server**, the command can choose one of the machine's addresses at random to be removed from the FLDB. Because this can have unpredictable results, always specify an address with **-server** when using the **-rmaddr** option. In addition, the command fails if the address to be removed is the only address present for the machine in the FLDB.

  If this option is specified, do not specify the **-addaddr** or **-changeaddr** option.

-addaddr *address*
: Adds the additional address specified with this option to the server entry specified by **-server** in the FLDB. A machine can have from one to four addresses associated with it in the FLDB. The command fails if you attempt to add a fifth address for the machine to the FLDB.

  If the name of the machine rather than one of its addresses is specified with **-server**, the command can choose one of the machine's addresses in the FLDB at random to have the address added to it. Because this can have unpredictable results, always specify an address with **-server** when using the **-addaddr** option.

  If this option is specified, do not specify the **-rmaddr** or **-changeaddr** option.

-changeaddr *address*
: Substitutes the address specified with this option for the address specified by **-server** in the FLDB. If the name of the machine rather than one of its addresses is specified with **-server**, the command can choose one of the machine's addresses at random to be replaced with the address specified with this option. Because this can produce unpredictable results, always specify an address with **-server** when using the **-changeaddr** option.

  If this option is specified, do not specify the **-rmaddr** or **-addaddr** option.

-principal *name*
Changes the abbreviation for the DFS server principal that is registered for the machine in the FLDB (for example, **hosts/** *hostname*). The machine's principal name in the Registry Database must match this name. If this option is omitted, the abbreviated DFS server principal currently associated with the server entry remains unchanged.

-quota *entries*
Changes the limit on the number of fileset entries (read/write, read-only, and backup) in the FLDB that can be associated with the specified **-server**. A value of 0 (zero) allows an unlimited number of fileset entries to be associated with the server. If this option is omitted, the number of fileset entries currently allowed for the specified File Server machine remains unchanged.

-owner *group*
Changes the group that is the owner of the server entry. In the entry, the specified group replaces the current owning group, if any. A group can be specified by a full or abbreviated group name (for example, ***/.../*** *cellname****/*** *group_name* or just *group_name*). Foreign groups cannot own a local server entry. If this option is omitted, no group owns the server entry. (The value **<nil>** is reported as the owner.) Use this option or use the **-noowner** option; omit both options to leave the current owning group unchanged.

-noowner
Specifies that no group is to own the server entry. In the entry, the empty group ID, displayed as **<nil>**, replaces the group that currently owns the server entry; the entry is unchanged in this regard if no group presently owns the server entry. Use this option or use the **-owner** option; omit both options to leave the current owning group unchanged.

-cell *cellname*
Specifies the cell in whose FLDB the server entry is to be modified. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts edserverentry** command alters a server entry in the FLDB for the server machine specified with the **-server** option. Use the **-rmaddr** option to remove an

address associated with a server from the FLDB. Use the -addaddr option to add a new address for a server to the FLDB, or use the -changeaddr option to change an address for a server in the FLDB.

The **-principal** option can be used to change the abbreviated DFS server principal associated with the server entry. The **-quota** option can be used to alter the number of fileset entries that can be associated with the File Server machine in the FLDB, and the **-owner** option can be used to assign a new group as the owner of the server entry (or the **-noowner** option can be used to indicate that no group owns the server entry).

Unless a value associated with a server entry is explicitly modified with this command, its current value in the FLDB remains unchanged. The values associated with a server entry are initially specified when the server entry is created with the **fts crserverentry** command. The values can then be modified at any time with the **fts edserverentry** command. Use the **fts lsserverentry** command to display the current values from the FLDB for a server entry. Use the **fts delserverentry** command to remove a server entry from the FLDB.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines.

EXAMPLES

The following command modifies the server entry in the FLDB for a server machine. The command changes the machine's network address from **191.54.206.36**, as specified with the **-server** option, to **191.54.206.46**, as indicated with the **-changeaddr** option. The command also allows the server to accommodate an unlimited number of fileset entries by providing a value of **0** (zero) with the **-quota** option.

```
$ fts edserverentry 191.54.206.36 -changeaddr 191.54.206.46 -quota 0
```

RELATED INFORMATION

Commands: **fts crserverentry(8dfs)**, **fts delserverentry(8dfs)**, **fts lsserverentry(8dfs)**.

fts help

Purpose

Shows syntax of specified **fts** commands or lists functional descriptions of all **fts** commands

Synopsis

fts help [**-topic** *string*]...[**-help**]

OPTIONS

-topic *string*
Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **lsft**, not **fts lsft**). If this option is omitted, the output provides a short description of all **fts** commands.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts help** command displays the first line (name and short description) of the online help entry for every **fts** command if -topic is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **fts apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

OUTPUT

The online help entry for each **fts** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with **Usage:**, lists the command options in the prescribed order.

EXAMPLES

The following command displays the online help entry for the **fts delmount** command:

```
$ fts help delmount
fts delmount: remove mount point
Usage: fts delmount -dir <directory_name>... [-help]
```

RELATED INFORMATION

Commands: **fts apropos(8dfs)**.

fts lock

Purpose

Locks a fileset entry in the FLDB

Synopsis

fts lock -fileset { *name* | *ID*}[**-cell** *cellname*][**-noauth** | **-localauth**] [**-verbose**] [**-help**]

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the fileset whose entry in the Fileset Location Database (FLDB) is to be locked. All versions of the fileset referenced in the entry are affected by the lock, regardless of whether the read/write, read-only, or backup version of the fileset is specified.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lock** command locks the entry in the FLDB for the fileset indicated with the **-fileset** option. Locking a fileset's FLDB entry blocks operations on all versions of the fileset, regardless of whether the read/write, read-only, or backup version of the fileset is indicated with the **-fileset** option. Locking a fileset's entry prevents all versions of the fileset from being modified with **fts** commands.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be locked resides.

CAUTIONS

Do not use this command in normal circumstances. It is useful only if the system administrator wants to guarantee that no one else manipulates the fileset until the lock is released and if there is reason to believe that locking will not happen automatically. Locking a fileset entry inhibits only operations such as deleting and cloning of the fileset; it does not prevent the reading of data from the fileset.

EXAMPLES

The following command locks the FLDB entry for *user.terry*:

```
$ fts lock user.terry
```

RELATED INFORMATION

Commands: **fts unlock(8dfs)**, **fts unlockfldb(8dfs)**.

fts lsaggr

Purpose

Lists all exported aggregates and partitions on a File Server machine

Synopsis

fts lsaggr -server *machine*[**-cell** *cellname*][**-noauth** | **-localauth][-verbose]** [**-help**]

OPTIONS

-server *machine*
Names the File Server machine whose exported aggregates and partitions are to be listed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsaggr** command displays information about all exported aggregates and partitions on the File Server machine specified by the **-server** option. The information about each aggregate and partition is specified in the */dcelocal***/var/dfs/dfstab** file on the machine.

You can also issue the **dfsexport** command with no options to list all aggregates and partitions currently exported from the local disk to the DCE namespace. You can use the **fts aggrinfo** command to display information about the amount of disk space available on a specific aggregate or partition or on all aggregates and partitions on a File Server machine.

Privilege Required

No privileges are required.

OUTPUT

This command displays a separate line for each aggregate or partition. Each line displays the following information:

- The aggregate name, specified in the second field of the **dfstab** file
- The device name, specified in the first field of the **dfstab** file
- The aggregate ID, specified in the fourth field of the **dfstab** file
- The file system type, specified in the third field of the **dfstab** file

EXAMPLES

The following example shows that two non-LFS partitions and two DCE LFS aggregates are exported from the File Server machine named */.../abc.com/hosts/fs1*:

```
$ fts lsaggr /.../abc.com/hosts/fs1
There are 4 aggregates on the server /.../abc.com/hosts/fs1 (fs1.abc.com):
             /usr (/dev/lv02): id=3     (non-LFS)
             /tmp (/dev/lv03): id=4     (non-LFS)
             lfs1 (/dev/lfs1): id=10     (LFS)
             lfs2 (/dev/lfs2): id=11     (LFS)
```

RELATED INFORMATION

Commands: **dfsexport(8dfs)**, **fts aggrinfo(8dfs)**.

Files: **dfstab(4dfs)**.

fts lsfldb

Purpose

Shows information from fileset entries in the FLDB

Synopsis

fts lsfldb[**-fileset** { *name* | *ID*}][**-server** *machine*][**-aggregate** *name*][**-locked**]
[**-cell** *cellname*][**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of a fileset about which information from the Fileset Location Database (FLDB) is to be displayed. Use this option or use -**server** (and optionally **-aggregate**), **-locked**, or both. Omit this option and the **-server** , **-aggregate**, and **-locked** options to display information about all fileset entries in the FLDB.

-server *machine*
Names a File Server machine about whose filesets information from the FLDB is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. This option can be combined with **-aggregate** to display information about the filesets on a single aggregate on **-server**, or it can be combined with **-locked** to display information about the filesets with locked FLDB entries on the server machine. Use this option alone or with **-aggregate**, **-locked**, or both, or use **-fileset**. Omit this option and the **-fileset**, **-aggregate**, and **-locked** options to display information about all fileset entries in the FLDB.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** about whose filesets information from the FLDB is to be displayed. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file. The **-server** option must be provided with this option. The **-locked** option can be supplied with this option to display information about the filesets with locked FLDB entries on the aggregate.

-locked
Specifies that the output show information only for filesets with locked FLDB entries. Use this option alone to see information for all filesets with locked FLDB entries. Use this option with **-server** (and optionally **-aggregate**) to see all filesets on a specific server machine (and optionally aggregate) with locked FLDB entries. Use this option alone or with **-server** (and optionally -aggregate) or use **-fileset**. Omit this option and the **-fileset**, **-server**, and **-aggregate** options to display information about all fileset entries in the FLDB.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsfldb** command formats and displays information about fileset entries from the FLDB. Its options can be combined to display information about a variety of different filesets. To display FLDB information for

- Every fileset entry, specify no options.
- Every fileset entry that mentions a specific File Server machine as the site of any version of a fileset, specify the name of the machine with **-server**.
- Every fileset entry that mentions a specific aggregate on a specific File Server machine as the site of any version of a fileset, specify both **-server** and **-aggregate**.
- The FLDB entries for filesets with locked entries, specify the **-locked** option alone or with **-server** (and optionally **-aggregate**).
- The fileset entry for a single fileset, specify the fileset name or ID number with **-fileset**.

Use the **fts lsheader** command to display information from fileset headers. To display more information about a single fileset, use the **fts lsft** command to display all of the information displayed by the **fts lsheader** command when the **-long** option is used and all of the information displayed by this command.

Privilege Required

No privileges are required.

OUTPUT

The **fts lsfldb** command displays the following information from the FLDB for each DCE LFS fileset specified with **-fileset** or **-server** (and optionally **-aggregate**). Because functionality such as replication is not supported for non-LFS filesets, this command displays less information for non-LFS filesets.

- The fileset's name.
- The fileset IDs of the read/write, read-only, and backup versions of the fileset.

- For each version, a status flag of **valid** indicates the version actually exists at a site; a status flag of **invalid** indicates the version does not exist at any site. (For the read-only version, the status flag indicates whether a replication site is defined.)
- The number of sites at which a version of the fileset exists.
- The maximum and minimum advisory RPC authentication bounds for use in communications with Cache Managers. There are two sets of bounds: One set governs communications with Cache Managers in the local cell, while the other set governs communications with Cache Managers in foreign cells. Currently, these bounds are not enforced but serve to bias the Cache Manager's initial authentication level.
- An indicator if the FLDB entry is locked. (The indicator is omitted if the entry is not locked.)
- The replication parameters associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) where read/write (RW), read-only (RO), or backup (BK) versions of the fileset reside.
- For a read-only version, the MaxSiteAge replication parameter defined for that site; for a read/write version, **0:00:00**.
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine (or **<nil>** if no group owns the server entry).

If the output includes more than one FLDB entry, information about the filesets is displayed in alphabetical order by fileset name. The last line of the output displays the total number of entries successfully reported and the total number of entries not reported (the number of entries that **failed**).

EXAMPLES

The following command shows an example of the output from the **fts lsfldb** command for a fileset named *user.terry*:

```
$ fts lsfldb user.terry
user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
Minimum local protection level: rpc_c_protect_level_none
Maximum local protection level: rpc_c_protect_level_pkt_privacy
Minimum remote protection level: rpc_c_protect_level_none
Maximum remote protection level: rpc_c_protect_level_pkt_privacy
number of sites: 1
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00;
  reclaimWait=18:00:00; minRepDelay=0:05:00; defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

RELATED INFORMATION

Commands: **fts lock(8dfs)**, **fts lsfldb(8dfs)**, **fts lsft(8dfs)**, **fts unlock(8dfs)**, **fts unlockfldb(8dfs)**.

Files: **dfstab(4dfs)**.

fts lsft

Purpose

Lists fileset information from both the fileset header and the FLDB entry

Synopsis

fts lsft {**-path** { *filename* | *directory_name*} | **-fileset** { *name* | *ID*}}[**-server** *machine*] [**-cell** *cellname*][**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-path { *filename* **|** *directory_name***}**
Names a file or directory on the fileset whose fileset header and FLDB information is to be displayed. Use this option or use **-fileset** ; omit both options to display information about the fileset that contains the current working directory.

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the fileset to be examined. The read/write, read-only, or backup version of the fileset can be specified. Append the **.backup** or **.readonly** extension to the name of the fileset to list information about the backup or read-only version instead of the read/write version; if the read/write version no longer exists, the command fails if the **.backup** or **.readonly** extension is not used with the name of the fileset.

Use this option or use **-path** ; omit both options to display information about the fileset that contains the current working directory.

-server *machine*
Names the File Server machine that houses the fileset about which information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

This option is useful for displaying information about a particular read-only replica of a DCE LFS fileset for which multiple replicas exist. If you include the **.readonly** extension with the name of a fileset specified with the **-fileset** option, specify the ID number of the read-only version of a fileset with the **-fileset** option, or specify the path to a file or directory in a read-only fileset with the **-path** option, you can use the **-server** option to indicate the machine that houses the specific replica about which information is to be displayed. If you omit the **-server** option in these cases, the command displays information about the replica at the fileset's oldest read-only site (the replica at the site that has been defined for the longest time).

This option is always unnecessary if information is to be displayed about the read/write or backup version of a fileset.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsft** command displays information from both the fileset header and the Fileset Location Database (FLDB) entry for the specified fileset. It displays the same output as the **fts lsheader** command with the **-long** option and the **fts lsfldb** command for a single fileset. It can be used to learn the fileset ID number of a fileset or to examine locked FLDB entries.

The fileset whose information is to be displayed can be specified by indicating the name of a file or directory on the fileset with the **-path** option, or it can be specified by indicating its name or ID number with the **-fileset** option. Omit both the **-path** and **-fileset** options to display information about the fileset that contains the current working directory. If the name of the fileset is specified with the **-fileset** option, the **.backup** or **.readonly** extension can be appended to the name to display information about one of those fileset versions rather than the read/write version.

If information about a read-only replica of a DCE LFS fileset is to be displayed and multiple replicas of the fileset exist, the **-server** option can be used to name the File Server machine that houses the specific replica about which information is to be displayed. Indicating a specific replica can be useful if network or hardware problems caused only some of a fileset's replicas to be updated. (By default, all replicas of the same fileset should always contain the same information.)

Use the **fts lsheader** command to display information from fileset headers. Use the **fts lsfldb** command to display information from fileset entries in the FLDB.

Privilege Required

No privileges are required.

OUTPUT

The **fts lsft** command displays the following information from the fileset header and the FLDB entry for a specified DCE LFS fileset. Because non-LFS filesets do not have DCE LFS fileset headers, and because functionality such as replication is not supported for non-LFS filesets, this command displays less information for a non-LFS fileset.

The command displays the following information from the fileset's header:

- The fileset's name (with a **.readonly** or **.backup** extension, if appropriate)
- Its fileset ID number

- Its type (**RW** for read/write, **RO** for read-only, or **BK** for backup)
- Its type (**LFS** or **non-LFS**)
- Information about the state of the fileset
- Its status (**On-line**, **Off-line**, or an error indicator)
- The File Server machine, aggregate name, and aggregate ID number on which it resides
- The ID numbers of the parent, clone, and backup filesets related to the fileset
- The ID numbers of the low-level backing and low-level forward filesets related to the fileset
- Its version number
- Its allocation and allocation usage, in kilobytes
- Its quota and quota usage, in kilobytes
- The day, date, and time when the fileset was created (replicated or backed up for a read-only or backup fileset)
- The day, date, and time when the contents of the fileset were last updated (same as the creation time for a read-only or backup fileset)

It then displays the following information from the fileset's entry in the FLDB:

- The fileset's name.
- The fileset IDs of the read/write, read-only, and backup versions of the fileset.
- For each version, a status flag of **valid** indicates the version actually exists at a site; a status flag of **invalid** indicates the version does not exist at any site. (For the read-only version, the status flag indicates whether a replication site is defined.)
- The maximum and minimum advisory RPC authentication bounds for use in communications with Cache Managers. There are two sets of bounds: One set governs communications with Cache Managers in the local cell while, the other set governs communications with Cache Managers in foreign cells. Currently, these bounds are not enforced but serve to bias the Cache Manager's initial authentication level.
- The number of sites at which a version of the fileset exists.
- An indicator if the FLDB entry is locked. (The indicator is omitted if the entry is not locked.)
- The replication parameters associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) on which read/write (**RW**), read-only (**RO**), or backup (**BK**) versions of the fileset reside.
- For a read-only version, the MaxSiteAge replication parameter defined for that site; for a read/write version, **0:00:00**.
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine (or **<nil>** if no group owns the server entry).

EXAMPLES

The following example displays information from the fileset header and FLDB entry for a DCE LFS fileset named *user.terry*:

```
$ fts lsft -fileset user.terry

_____________________________________________________
user.terry 0,,196953 RW LFS     states
0x10010005 On-line
    fs3.abc.com, aggregate
```

```
lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    429496729 K alloc limit;     1252 K alloc usage
        15000 K quota limit;     9340 K quota usage
    Creation Fri Oct 15 16:45:16 1993
    Last Update Mon Nov 22 11:36:00 1993


user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
Minimum local protection level: rpc_c_protect_level_none
Maximum local protection level: rpc_c_protect_level_pkt_privacy
Minimum remote protection level: rpc_c_protect_level_none
Maximum remote protection level: rpc_c_protect_level_pkt_privacy
number of sites: 2
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00;
  reclaimWait=18:00:00; minRepDelay=0:05:00;
defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

RELATED INFORMATION

Commands: **fts lsfldb(8dfs)**, **fts lsheader(8dfs)**.

fts lsheader

Purpose

Shows information from fileset headers

Synopsis

fts lsheader -server *machine*[**-aggregate** *name*]{**-fast** | **-long** }[**-cell** *cellname*] [{**-noauth** | **localauth** }][**-verbose**][**-help**]

OPTIONS

-server *machine*
Names a File Server machine about whose filesets header information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. This option can be combined with the **-aggregate** option to name a specific aggregate on -server.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** from whose filesets header information is to be displayed. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file. The **-server** option must be provided with this option.

-fast Directs the output to display only the fileset ID numbers of all filesets on the indicated server (and optionally the aggregate). If you use this option, do not use the **-long** option.

-long Directs the output to display more detailed information about all filesets on the indicated server (and optionally the aggregate). If you use this option, do not use the **-fast** option.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsheader** command formats and displays information from the fileset headers of filesets on the specified server (and optionally the aggregate or partition). To display information from the headers of all filesets on a specific File Server machine, specify the name of the server machine with the **-server** option. To specify information from the headers of all filesets on a specific aggregate on a File Server machine, specify the name of the server machine with the **-server** option and the name of the aggregate or partition with the **-aggregate** option.

Include the **-fast** option with the command to display only the ID numbers of the filesets at the specified location. Include the **-long** option with the command to display more detailed information from the headers of the filesets at the specified location.

Use the **fts lsfldb** command to display information from fileset entries in the Fileset Location Database (FLDB). To display more information about a single fileset, use the **fts lsft** command to display all of the information displayed by this command when the **-long** option is used and all of the information displayed by the **fts lsfldb** command.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine specified by **-server**.

OUTPUT

The **fts lsheader** command displays different output about the filesets at the specified location depending on whether the **-fast** or **-long** option is included. Information about the filesets is displayed in numeric order by fileset ID number if the **-fast** option is used; otherwise, it is displayed in alphabetical order by fileset name.

The information described in this section is displayed for DCE LFS filesets. Because non-LFS filesets do not have DCE LFS fileset headers, the **fts lsheader** command displays much less information for non-LFS filesets, and the **-fast** and **-long** options have less of an impact on the amount of output displayed.

If the **-fast** option is used, the command lists the ID number of each fileset. If the **-aggregate** option is omitted, the command also displays the total number of filesets on the specified server.

If both the **-fast** and **-long** options are omitted, the command displays the following information:

- The File Server machine, aggregate name, and aggregate ID number where the filesets reside.
- The total number of filesets on the aggregate.
- Each fileset's name (with a **.readonly** or **.backup** extension, if appropriate).
- Each fileset's fileset ID number.
- Each fileset's type (**RW** for read/write, **RO** for read-only, or **BK** for backup).
- Each fileset's allocation usage and quota usage, in kilobytes.
- Each fileset's status (**On-line**, **Off-line**, or an error indicator).
- The total number of filesets online, the total number of filesets offline, and the total number of filesets busy. A busy fileset is one upon which a fileset-related

operation is currently in progress (for example, the fileset is being moved or cloned, or the Replication Server is currently forwarding changes from the fileset to read-only replicas).

If the **-long** option is used, the command displays the following additional information for each fileset:

- Whether it is a DCE LFS (**LFS**) or **non-LFS** fileset
- Information about the state of the fileset
- The ID numbers of the parent, clone, and backup filesets related to the fileset
- The ID numbers of the low-level backing and low-level forward filesets related to the fileset
- The version number of the fileset
- The allocation and allocation usage, in kilobytes, of the fileset
- The quota and quota usage, in kilobytes, of the fileset
- The day, date, and time when the fileset was created (replicated or backed up for a read-only or backup fileset)
- The day, date, and time when the contents of the fileset were last updated (same as the creation time for a read-only or backup fileset)

EXAMPLES

The following examples show output from the **fts lsheader** command when it is executed with the **-fast** option, with neither the **-fast** option nor the **-long** option, and with the **-long**option. All three examples display output primarily for the same fileset, *user.terry* (ID number **0,,196953**).

The following examples show output from the **fts lsheader** command when it is executed with the **-fast** option, with neither the **-fast** option nor the **-long** option, and with the **-long** option. All three examples display output primarily for the same fileset, *user.terry* (ID number **0,,196953**).

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1  fast
0,,196953
0,,196956
    .
    .
0,,199845
0,,199846
Total number of filesets on server fs3: 16
$ fts lsheader /.../abc.com/hosts/fs3


Total filesets on server fs3 aggregate lfs1 (ID
10): 16
user.terry            0,,196953 RW   5071 K alloc  8421 K quota On-line
user.wvh              0,,196956 RW   4955 K alloc  9371 K quota On-line
    .
    .
Total filesets on-line 15; total off-line 1; total busy 0

$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1 -long
```

```
Total filesets on server fs3 aggregate lfs1 (ID 10): 16
user.terry 0,,196953 RW LFS     states 0x10010005   On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    429496729 K alloc limit;     1252 K alloc usage
        15000 K quota limit;     9340 K quota usage
    Creation Tue Oct 15 16:45:16 1991
    Last Update Fri Nov 22 11:36:00 1991
  user.wvh 0,,196956 RW LFS     states 0x10010005  On-line
    .
    .
Total filesets on-line 15; total off-line 1; total busy 0
```

RELATED INFORMATION

Commands: **fts lsfldb(8dfs)**, **fts lsft(8dfs)**.

Files: **dfstab(4dfs)**.

fts lsmount

Purpose

Lists the filesets associated with mount points

Synopsis

fts lsmount -dir *directory_name*[**-help**]

OPTIONS

-dir *directory_name*
Names each directory that serves as a mount point for a fileset. The last element in the specified pathname must be an actual name, not. (dot) or.. (dot dot).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsmount** command displays the name of the fileset for which each directory specified with the **-dir** option is the mount point. The association between a mount point and a fileset is created with the **fts crmount** command; it is removed with the **fts delmount** command.

Privilege Required

The issuer must have read permission on each directory indicated with the -dir option, regardless of whether each indicated directory resides in a directory in a DCE LFS or non-LFS fileset.

OUTPUT

The **fts lsmount** command displays the following message for each directory that is a mount point:

'directory_name' is a mount point for fileset *'fileset_name*'

where *directory_name* is the name of a directory specified with the **-dir** option, and *fileset_name* is the name of the fileset for which *directory_name* serves as a mount point. The command also provides the following information about the directory and fileset:

 # (number sign)
Precedes *fileset_name* if *directory_name* is a regular mount point.

% (percent sign)
Precedes *fileset_name* if *directory_name* is a read/write mount point.

! (exclamation point)
Replaces *fileset_name* if the directory is a global root mount point (a mount point for the root of the DCE global namespace).

The **fts lsmount** command displays the following message for each directory that is not a mount point:

'directory_name' is not a mount point.

EXAMPLES

The following example lists the mount point *vijay*, which is a regular mount point for the fileset named *user.vijay*:

```
$ fts lsmount vijay
'vijay' is a mount point for fileset '#user.vijay'
```

RELATED INFORMATION

Commands: **fts crmount(8dfs)**, **fts delmount(8dfs)**.

fts lsquota

Purpose

Shows quota and quota usage information for filesets and disk size and usage information for aggregates or partitions

Synopsis

fts lsquota {**-path** { *filename* | *directory_name*...}**-fileset** {*name* | *ID*}} [**-cell** *cellname*][**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-path *filename* **or** *directory_name*
Names a file or directory from each fileset about which quota, size, and usage information is to be displayed. Include filenames or directory names from different filesets if desired. It is not necessary to name more than one file or directory from the same fileset. Use this option or use -fileset ; omit both options to display information about the fileset containing the current working directory.

-fileset *name* **or** *ID*
Specifies the complete name or fileset ID number of each fileset about which quota, size, and usage information is to be displayed. Use this option or use **-path** ; omit both options to display information about the fileset that contains the current working directory.

-cell *cellname*
Specifies the cell with respect to which the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. Generally, the **-noauth** option is included if DFS authorization checking is disabled on a server machine on which administrative privilege is required or if the Security Service is unavailable. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions during command execution.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsquota** command displays quota and quota usage information about filesets and disk size and usage information about the partitions or aggregates on

which the filesets reside. Use the **-path** option to specify a file or directory on a fileset to see information about that fileset; use the **-fileset** option to specify the name or ID number of a fileset to see information about that fileset; omit both options to see information about the fileset that contains the current working directory.

For DCE LFS filesets, the **fts lsquota** command displays the quota and quota use (in kilobytes) and the percentage of the quota in use. For both DCE LFS and non-LFS filesets, this command displays the name of the fileset, information about the number of available kilobytes on the aggregate or partition on which the fileset resides, the number of kilobytes in use on the aggregate or partition, and the percentage of the aggregate or partition in use. It also reports whether the device is a DCE LFS aggregate or a non-LFS partition.

The size of a non-LFS fileset is equal to the size of the partition on which it resides. Therefore, the size and usage information displayed for the partition (non-LFS aggregate) in the output of the **fts lsquota** command equals the quota and quota usage information of the fileset on the partition. Using this command with a non-LFS fileset is analogous to using the UNIX **df** command with the partition on which the fileset resides. (Note that the **df** command can be used to display the size of exported DCE LFS aggregates and locally mounted DCE LFS filesets, but it cannot be used to display the size of a DCE LFS fileset that is not mounted locally.)

The **fts lsheader** and **fts lsft** commands can be used to display the quota of a DCE LFS fileset. The **fts aggrinfo** command can be used to display the total disk space on an aggregate and the amount currently available.

By default, every newly created DCE LFS fileset has a quota of 5000 kilobytes. The **fts setquota** command can be used to increase or decrease the quota of a DCE LFS fileset. Because the quota of a DCE LFS fileset does not represent the amount of physical data stored on the fileset, it can be larger than the size of the aggregate on which the fileset resides. Similarly, the combined quotas of all filesets on an aggregate can be larger than the size of the aggregate.

The quota of a non-LFS fileset cannot be changed via DFS. (The **fts setquota** command works only with DCE LFS filesets.)

Privilege Required

No privileges are required.

OUTPUT

This command displays the following information about each specified fileset:

- The name of the fileset
- The quota, in kilobytes, of the fileset (DCE LFS only)
- The number of kilobytes of the quota currently in use on the fileset (DCE LFS only)
- The percentage of the quota currently in use on the fileset (DCE LFS only)
- The percentage of available disk space currently in use on the aggregate or partition on which the fileset resides
- The number of kilobytes of disk space in use and available on the aggregate or partition on which the fileset resides
- The file system type of the aggregate (**LFS** or **non-LFS**)

If the fileset quota usage rises above 90% or the aggregate or partition usage rises above 97%, the appropriate percentage is indicated with **<<** and the message **<<WARNING** is displayed at the end of the output line.

Note: Because each non-LFS partition contains a single fileset, the information displayed for a non-LFS partition applies to the single non-LFS fileset it houses. Ignore the quota, quota usage, and quota usage percentage values displayed for a non-LFS fileset; they are always 0 (zeros). Consult the disk size, usage, and percentage values displayed for the partition on which the non-LFS fileset resides to determine the corresponding values for the fileset.

EXAMPLES

The following command lists quota and quota usage information for the fileset that contains the directory named */.../abc.com/fs/usr/terry*, and it displays size and usage information for the aggregate that contains this fileset. The command also displays size and usage information for the partition that contains the directory named */.../abc.com/fs/usr/jlw*. The first directory resides on the DCE LFS fileset named *user.terry*; the quota of the DCE LFS fileset is less than the size of the aggregate on which it is located. The second directory resides on the non-LFS fileset named *user.jlw*; the quota of the non-LFS fileset is the same as the size of the partition on which it is located.

```
$ fts lsq /.../abc.com/fs/usr/terry /.../abc.com/fs/usr/jlw


Fileset Name   Quota   Used   % Used   Aggregate
user.terry     15000   5071      34%     86% = 84538/98300 (LFS)
user.jlw           0      0       0%     84% = 8448/10000 (non-LFS)
```

The following command lists quota and usage information for the DCE LFS fileset named *user.jean*, and size and usage information for the aggregate on which the fileset resides. The **<<WARNING** message directs the issuer's attention to the fact that the percentage of the quota in use on the indicated fileset is well above the warning level of 90%.

```
$ fts lsq -f user.jean
Fileset Name   Quota   Used   % Used   Aggregate
user.jean       5000   4955      99%<<   92% = 87436/98300 (LFS) <<WARNING
```

RELATED INFORMATION

Commands: **fts aggrinfo(8dfs)**, **fts lsft(8dfs)**, **fts lsheader(8dfs)**, **fts setquota(8dfs)**.

fts lsreplicas

Purpose

Displays the statuses of DCE LFS fileset replicas

Synopsis

```
fts lsreplicas -fileset { name | ID}{-all | -server machine}[-cell cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* | *ID*}
Specifies the complete name or fileset ID number of the fileset whose replicas are to be checked.

-all Specifies that all replicas of **-fileset** are to be checked. Use this option or use **-server**.

-server *machine*
Names a specific File Server machine on which replicas of **-fileset** are to be checked. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option or use **-all**.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsreplicas** command shows the replication statuses of read-only replicas of the read/write DCE LFS fileset specified with the **-fileset** option. Use the command's options to check replicas of **-fileset** as follows:

- To check the status of the replica stored on a specific File Server machine, specify the name of the machine with the **-server** option.
- To check the status of all replicas, specify the **-all** option.

If Release Replication is used, the read-only replica at the replica site defined on the same file server machine as the source fileset is not listed.

If Release Replication is used for a read/write fileset, use the **fts release** command to place replicas of the fileset at replication sites. (If Scheduled Replication is used, the Replication Server automatically places replicas at replication sites according to specified parameters.) Use the **fts update** command to request that the Replication Server make an immediate update of the replicas of any read/write fileset.

Use the **fts statrepserver** command to check the status of the Replication Server process on a specific File Server machine. Use the **fts addsite** command to add a replication site; use the **fts rmsite** command to remove a replication site.

Privilege Required

No privileges are required.

EXAMPLES

The following command displays the status of each replica of the read/write fileset named **rs_aix41.bin**:

```
$ fts lsr rs_aix41.bin -a
```

RELATED INFORMATION

Commands: **fts addsite(8dfs)**, **fts release(8dfs)**, **fts rmsite(8dfs)**, **fts statrepserver(8dfs)**, **fts update(8dfs)**.

fts lsserverentry

Purpose

Lists a server entry from the FLDB

Synopsis

```
fts lsserverentry -server{ machine | -all }[-cell cellname]
[-noauth | -localauth ][-verbose ]
[-help ]
```

OPTIONS

-server *machine*
Specifies the name of the server machine whose entry in the Fileset Location Database (FLDB) is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option or use the **-all** option.

-all Specifies that the entries for all server machines in the FLDB are to be displayed. Use this option or use the **-server** option.

-cell *cellname*
Specifies the cell from whose FLDB the specified server entries are to be listed. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts lsserverentry** command displays server entry information from the FLDB. If the **-server** option is specified, entry information from the FLDB for only the indicated server machine is displayed. If the **-all** option is specified, entry information from the FLDB for all server machines is displayed.

Use the **fts crserverentry** command to create a server entry in the FLDB. Use the **fts edserverentry** command to modify a server entry in the FLDB. Use the **fts delserverentry** command to remove a server entry from the FLDB.

Privilege Required

No privileges are required.

EXAMPLES

The following command displays the server entry from the FLDB for a server machine named **fs1**:

```
$ fts lsserverentry /.../abc.com/hosts/fs1
```

RELATED INFORMATION

Commands: **fts crserverentry(8dfs)**, **fts delserverentry(8dfs)**, **fts edserverentry(8dfs)**.

fts move

Purpose

Moves a read/write DCE LFS fileset to another site

Synopsis

```
fts move -fileset { name | ID} -fromserver  source_machine -fromaggregate
 source_name -toserver  dest_machine -toaggregate  dest_name [-cell  cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* | *ID*}
Specifies the complete name or the fileset ID number of the read/write fileset to be moved.

-fromserver *source_machine*
Names the File Server machine on which the fileset currently resides. Specify the File Server machine by its DCE pathname, its host name, or its IP address.

-fromaggregate *source_name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate on which the fileset currently resides. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file.

-toserver *dest_machine*
Names the File Server machine to which the fileset is to be moved. Specify the File Server machine by its DCE pathname, its host name, or its IP address.

-toaggregate *dest_name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate to which the fileset is to be moved. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **dfstab** file.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts move** command moves the indicated read/write DCE LFS fileset from its current site (specified with the **-fromserver** and **-fromaggregate** options) to the destination site (specified with the **-toserver** and **-toaggregate** options). The command decrements the number of fileset entries recorded as residing on the machine indicated with the **-fromserver** option in the Fileset Location Database (FLDB) entry for the machine, and it increments the number of fileset entries recorded as residing on the machine specified with the -toserver option in the FLDB entry for that machine. It also automatically removes the backup version of the fileset, if it exists, from the current site. To create a new backup version at the destination site, use the **fts clone** command.

If the fileset to be moved uses Scheduled Replication, the command has no effect on the fileset's replicas. However, if the fileset to be moved uses Release Replication, the command has the following effects:

- If the fileset is moved to a different File Server machine, the command removes the replication site and replica that resides at the fileset's current site. It then creates a new replication site and replica on the File Server machine and aggregate to which the fileset is moved. The new replica is created as a clone of the read/write fileset. If the machine to which the fileset is to be moved already houses a replication site and replica of the fileset, you must use the **fts rmsite** command to remove the existing replication site and replica from the destination machine before issuing the **fts move** command.
- If the fileset is moved to a different aggregate on the same File Server machine, the command does one of the following:
  - If the replica resides on the aggregate from which the fileset is moved, the command deletes the existing replication site and replica. It then creates a new replication site and replica on the aggregate to which the fileset is moved, creating the new replica as a clone.
  - If the replica resides on the aggregate to which the fileset is moved, the command has no immediate effect on the replica. However, the next time the **fts release** command is used, the replica is changed from a full read-only replica to a clone.
  - If the replica resides on neither of the aggregates involved in the move, the command does not affect the replica.

It is not possible to move a read-only or backup fileset. For read-only filesets, the corresponding action is to create a new replication site with the **fts addsite** command and remove an existing one with the **fts rmsite** command. Because the backup version of a read/write fileset is automatically deleted when its read/write source is moved, a backup fileset can be moved only by moving its read/write source fileset and issuing the **fts clone** command to create a new backup version.

Furthermore, it is not possible to move a fileset from a site in one cell to a site in another cell. Filesets can be moved only between two sites in the same cell. The filesets are assumed to reside in the local cell of the issuer unless the name of a foreign cell is specified with the **-cell** option.

A DCE LFS fileset that is mounted locally (as a file system on its File Server machine) cannot be moved to a different File Server machine. It can be moved only

to a different aggregate on the same File Server machine. If the command is used to move a DCE LFS fileset that is locally mounted, its -fromserver and **-toserver** options must name the same File Server machine; otherwise, the command fails. (To move a locally mounted fileset to a different machine, remove its local mount point before issuing this command.)

In addition, because the backup version of a fileset is removed when its read/write version is moved, you cannot move a fileset (not even to another aggregate on the same File Server machine) if its backup version is mounted locally. You must remove the backup version's local mount point before moving the fileset. If you issue the **fts move** command against the DCE LFS filesets that are replicated, the command may fail because of a conflict between the move operation and the propagation of updates to read-only replicas. Two types of failures can occur:

- If you issue the **fts move** command when a **repserver** process is propagating changes from the read/write fileset to a read-only fileset, the move command fails.
- If a **repserver** process attempts to propagate changes from the read/write fileset to a read-only replica while an **fts move** is in progress, the move might be interrupted and fail.

In both failures, the message displayed is the following:

```
Error in move: the repserver forwarding fileset (dfs / xvl)
```

To avoid this problem, verify that all the read-only replicas of the fileset match the read/write fileset prior to the move by comparing the fileset version numbers. Use the output from the **-long** option of the **fts lsheader** command for this comparison. Also ensure that no changes are made to the read/write fileset during the move operation. For filesets that use release replication, do not issue the **fts release** command immediately before or during an **fts move operation**.

Privilege Required

The issuer must be listed in the **admin.ft** files on both the source and destination machines. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entries for the source machine, the destination machine, and any machines on which replicas of the fileset reside. In addition, the source machine (-fromserver) must be listed in the **admin.ft** file on the destination machine (-toserver).

EXAMPLES

The following command moves the fileset *user.smith* from **/dev/lv01** on **fs3** to **/dev/lv02** on **fs7**:

```
$ fts move user.smith /.../abc.com/hosts/fs3 /dev/lv01/.../abc.com/hosts/fs7
/dev/lv02
```

RELATED INFORMATION

Commands: **fts addsite(8dfs)**, **fts clone(8dfs)**, **fts delete(8dfs)**, **fts release(8dfs)fts rmsite**.

Files: **dfstab(4dfs)**.

fts release

Purpose

Initiates Release Replication by placing read-only version of a read/write DCE LFS fileset at the local site

Synopsis

fts release -fileset { *name* | *ID*}[**-wait**][**-cell** *cellname*][**-noauth** | **-localauth**] [**-verbose**][**-help**]

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the read/write fileset to be replicated locally (cloned if the local replication site is defined on the same aggregate as the read/write fileset). Once the fileset is replicated locally, the Replication Servers at the fileset's replication sites copy the replica to their sites.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-wait Directs the command to not terminate (return a prompt) until all replicas are up to date. By default, the command returns immediately without waiting for the Replication Servers to complete propagation.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts release** command is used to initiate the replication process for a fileset that uses Release Replication. The command ″releases″ a new read-only copy of the DCE LFS fileset specified with the **-fileset** option. It places the new read-only copy at the local replication site defined on the same File Server machine as the read/write fileset. The Replication Servers at each of the fileset's replication sites (specified File Server machines and aggregates) then update the copies of the read-only replica at the sites on their respective machines.

Note that, as with updating a new version of a fileset that uses Scheduled Replication, releasing a fileset that uses Release Replication does not ensure immediate access to data in the new version of the replica. A Cache Manager continues to provide data cached from the old version of the replica until the MaxAge for the fileset expires or until the Cache Manager needs to access data from the replica that it has not already cached.

To gain immediate access to data in the new version of the replica, issue the **cm flush** or **cm flushfileset** command to flush the old data from the cache. This forces the Cache Manager to replace data it has cached from the replica. Replication Servers begin replication in parallel; however, until all replicas have been updated, you cannot directly force the Cache Manager to access data from the new version of the replica.

Before the **fts release** command can be used, the **fts setrepinfo** command must be used to define the replication parameters for the read/write fileset. If Release Replication is to be used, the **-release** option must be specified with the **fts setrepinfo** command. The **fts addsite** command must also be used to define the replication sites for the read/write fileset. For Release Replication, the replication site on the same File Server machine as the read/write fileset must be defined first. The read/write fileset must have at least one replication site defined before the **fts release** command can be issued. The replication parameters and sites for a read/write fileset are recorded in the fileset's entry in the Fileset Location Database (FLDB).

The **fts release** command does not alter the replication type and parameters defined for the specified fileset. The command can be used only with a fileset that uses Release Replication; it returns an error if the specified fileset uses Scheduled Replication. The **fts update** command can be used to request an immediate update of the replicas of a fileset that uses Scheduled Replication.

Use the **fts lsreplicas** command to check the status of replicas. Use the **fts statrepserver** command to check the status of the Replication Server on a File Server machine.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine on which the source read/write fileset is stored. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entries for the machine on which the source fileset resides and all machines on which the read-only replicas are to reside.

EXAMPLES

The following command releases (initiates Release Replication for) the fileset named **pmax_osf1.bin**:

```
$ fts release pmax_osf1.bin
```

RELATED INFORMATION

Commands: **cm flush(8dfs)**, **cm flushfileset(8dfs)**, **fts addsite(8dfs)**, **fts lsreplicas(8dfs)**, **fts setrepinfo(8dfs)**, **fts statrepserver(8dfs)**, **fts update(8dfs)**.

fts rename

Purpose

Renames a fileset

Synopsis

fts rename -oldname *oldname* **-newname** *newname* [**-cell** *cellname*]
[**-noauth** | **-localauth [-verbose][-help]**

OPTIONS

-oldname *oldname*
Specifies the current name of the read/write fileset.

-newname *newname*
Specifies the new name for the read/write fileset. The name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The. (dot)
- The – (dash)
- The _ (underscore)

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. It can be no longer than 102 characters. This length does not include the **.readonly** or **.backup** extension that is added automatically when a read-only or backup version of a DCE LFS fileset is created. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup DCE LFS filesets, so you cannot specify a fileset name that ends with either of these extensions.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts rename** command changes the name of the read/write fileset specified with **-oldname** to the name specified with **-newname**. The names of the read/write fileset's read-only copies and backup copy, if any, automatically change to match.

After issuing this command, the issuer must correct any mount points that refer to the old fileset name. This is done by removing each old mount point with the **fts delmount** command and creating a replacement for each with the **fts crmount** command. (These commands require that the issuer have the necessary file system permissions for the operations.)

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine on which the read/write fileset resides. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be renamed resides.

EXAMPLES

The following command changes the incorrect fileset name **osf1.bin** to the correct fileset name **pmax_osf1.bin**:

```
$ fts rename osf1.bin pmax_osf1.bin
```

RELATED INFORMATION

Commands: **fts crmount(8dfs)**, **fts delmount(8dfs)**.

fts restore

Purpose

Converts a dump file from bytestream format to fileset format and places it in the file system

Synopsis

fts restore -ftname *name* **-server** *machine* **-aggregate** *name*[**-file** *filename*][**-ftid** *ID*] [**-overwrite**][**-cell** *cellname*][**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-ftname *name*

Specifies the name of the fileset to which the file is to be restored. If the file is to be restored as a new fileset, the name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The. (dot)
- The – (dash)
- The _ (underscore)

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. It can be no longer than 102 characters. This length does not include the **.readonly** or **.backup** extension that is added automatically when a read-only or backup version of a DCE LFS fileset is created. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup DCE LFS filesets, so you cannot specify a fileset name that ends with either of these extensions.

-server *machine*

Specifies the File Server machine to which the file is to be restored. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*

Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** to which the file is to be restored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file.

-file *filename*

Specifies the complete pathname of the file to be restored. If a complete pathname is not provided, the file is assumed to reside in the current working directory. If this option is omitted, the data is read from standard input (**stdin**).

-ftid *ID*

Specifies the fileset ID number to assign to the restored fileset. If this option is omitted and an existing fileset is to be overwritten, the fileset ID number of the existing fileset is used. If it is omitted and a new fileset is to be

created, the FL Server allocates a new fileset ID number for the fileset. Use this option only when restoring a dump file as a DCE LFS fileset; use it sparingly and with great care.

-overwrite
Specifies that the file to be restored can overwrite an existing fileset. If this option is omitted, the command exits without overwriting an existing fileset. You must use this option to overwrite a previously restored version of a fileset with an incremental dump of the same fileset; more information about conditions that must be met if a fileset is to be overwritten by an incremental dump is provided later in this reference page. You must also use this option to restore a dump file as a non-LFS fileset.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts restore** command translates a dump file created previously with the **fts dump** command from a bytestream format to a fileset format appropriate for the machine specified with the **-server** option. The dump file to be restored is indicated with the **-file** option. If this option is omitted, the data to be restored is read from **stdin**.

The fileset contained in the dump file can be restored as a new read/write DCE LFS fileset by specifying a name and site for the new fileset. The command assigns the fileset the name indicated with the **-ftname** option. It restores it to the site specified with the **-server** and **-aggregate** options. The dump file must contain the full dump of a fileset if it is to be restored as a new fileset.

Alternatively, the fileset contained in the dump file can be restored over an existing read/write version of the same fileset by specifying the name and site of the existing fileset. The command resets the creation time stored in the fileset's header to match the restore time. The **-overwrite** option must be used to specify that the dump file is to overwrite the existing fileset. If this option is omitted, the command displays an error message and exits instead of overwriting the existing fileset.

When restoring a dump file as a non-LFS fileset, the fileset must already exist for the non-LFS partition on which it resides to be exported to the DCE namespace. In this case, you must use the **-overwrite** option to overwrite the existing non-LFS fileset (even if the fileset to be overwritten contains no data).

If you are overwriting an existing fileset with an incremental dump, the fileset to be overwritten should initially have been restored as a new read/write fileset from a full dump. Also, both the dump file to be restored and the full dump that initially produced the read/write fileset to be overwritten must be dumps of the same fileset. (A full dump of a fileset can be restored to overwrite an existing fileset, but the restored dump file overwrites all data in the existing fileset. An incremental dump of a fileset cannot be restored to overwrite an existing fileset that was not created from the restoration of a full dump.)

Multiple incremental dumps of a fileset can be restored to overwrite the same existing fileset provided the following conditions are true:

- The fileset to be overwritten must not have been modified (that is, no files added, removed, or saved, and no ACLs changed) since its most recent restoration from a full or incremental dump.
- The dump file to be restored must have been created *from* a date and time (as specified with the **-date** or **-version** option of the **fts dump** command) *no later* than the date and time at which the most recently restored dump of the fileset to be overwritten was dumped.
- The dump file to be restored must have been created *at* a date and time *later* than the date and time at which the most recently restored dump of the fileset to be overwritten was dumped.

The last two conditions indicate that the span of time recorded in the incremental dump to be restored must overlap and extend the span of time recorded in the fileset to be overwritten. For example, suppose the following dumps were made of a fileset: a full dump was made on 1 January 1992, an incremental dump from 31 December 1991 was made on 7 January 1992, and an incremental dump from 6 January 1992 was made on 14 January 1992. The following sequence of operations represents the only possible way to restore the fileset from all three of these dumps:

1. The full dump made on 1 January is restored as a new read/write fileset.
2. The incremental dump made on 7 January is restored to overwrite the read/write version of the fileset made from the full dump.
3. The incremental dump made on 14 January is restored to overwrite the read/write version of the fileset that includes data from the full and first incremental dumps.

No other sequence of restore operations involving all three dumps is possible. Any other sequence of steps will undoubtedly result in some or all of the data in the fileset being inaccessible or inconsistent.

When restoring a dump file as a DCE LFS fileset, a fileset ID number can be assigned to the restored fileset with the **-ftid** option. This is generally not recommended unless there is good reason to believe that an available fileset ID number can be specified. If the **-ftid** option is omitted, an overwritten DCE LFS fileset retains its current ID number, or the FL Server allocates a new ID number for a new DCE LFS fileset restored from a dump file. If a new fileset ID number is

assigned or allocated, the FL Server increments the number of fileset entries recorded as residing on the specified File Server machine in the Fileset Location Database (FLDB) entry for the server.

When restoring a dump file as a non-LFS fileset, do not use the **-ftid** option. Omit the option to continue to use the fileset ID number specified for the non-LFS fileset in the entry for its partition in the **dfstab** file. (Note that the restored dump file overwrites all data on the non-LFS partition.)

If a new fileset is created, use the **fts crmount** command to create a mount point for the fileset, making it visible in the DCE namespace. If an existing DCE LFS fileset is overwritten with this command, use the **fts update** command to release new read-only replicas based on the new version of the fileset, and use the **fts clone** command to create a new backup version of the fileset, as necessary.

Note: On DCE 2.2 for AIX, dump of non-LFS filesets is not supported.

You can use the **fts restore** command to restore a dump file to any type of fileset (DCE LFS or non-LFS), regardless of the type of fileset from which it was created. For example, a dump file of a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of non-LFS fileset. Similarly, a dump file of a non-LFS fileset can be restored to a DCE LFS fileset or to a different type of non-LFS fileset. In any case, the contents of the dump file are translated into the appropriate format for the file system to which they are restored. (Refer to your vendor's documentation to verify the level of support for dump and restore operations between different types of file systems.)

Note that incompatible information may be lost when a fileset is dumped and restored between different types of file systems. For example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs.

You cannot restore a fileset dumped in one cell to a site in another cell.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine specified by **-server** and must have the read permission on the dump file. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset is recorded as residing in the FLDB (generally only **-server** unless an existing fileset is to be overwritten).

CAUTIONS

Ensure that all of the conditions discussed in the description section are met before restoring an incremental dump of a fileset over an existing fileset. Violation of any of the conditions is very likely to result in inaccessibility or inconsistency of some or all of the data in the fileset.

EXAMPLES

The following example restores a file, */tmp/smith.013191.dump*, that contains an incremental dump of a fileset over an existing read/write version of the same fileset, *user.smith*. The incremental dump was created using a start date and time no later than the date and time when the most recently restored version of the fileset to be overwritten was dumped, and it was dumped at a date and time later than the date

and time when the most recently restored version of the fileset to be overwritten was dumped. Also, the fileset to be overwritten has not been modified since it was last restored. The **-ftid** option is omitted, so the fileset retains its current fileset ID number.

```
$ fts restore user.smith /.../abc.com/hosts/fs1 lfs1/tmp/smith.013191.dump
-overwrite
```

The following command takes input directly from an **fts dump** command to create a new read/write fileset, *user.terry*, from an existing fileset, *user.smith*. The **-file** option is omitted from the **fts dump** command to send the output to **stdout**, and it is omitted from the **fts restore** command to read the input from **stdin**. (The information is ″piped″ from one command to the next.) The -ftid option is again omitted from the **fts restore** command; this time the FL Server allocates a new ID number for the fileset.

```
$ fts dump user.smith -time 0 | fts restore user.terry /.../abc.com/hosts/fs1
lfs1
```

RELATED INFORMATION

Commands: **fts clone(8dfs)**, **fts crmount(8dfs)**, **fts dump(8dfs)**, **fts update(8dfs)**.

Files: **dfstab(4dfs)**.

fts rmsite

Purpose

Removes a replication site and read-only DCE LFS fileset

Synopsis

```
fts rmsite  { name | ID -server machine -aggregate name[-cell cellname]
[{-noauth | -localauth }[-verbose ][-help ]
```

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the read/write fileset for which a replication site and the read-only fileset stored at that site are to be removed.

-server *machine*
Specifies the File Server machine to be removed as a replication site. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate to be removed as a replication site. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the */dcelocal***/var/dfs/dfstab** file. If the aggregate is not currently exported or has been detached, you must specify the aggregate ID.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts rmsite** command removes a replication site currently defined for the read/write DCE LFS fileset specified with the **-fileset** option. The -server and **-aggregate** options are used to specify the replication site to be removed. The command performs the following actions:

- It removes the definition of the replication site from the Fileset Location Database (FLDB) entry for the fileset.
- It decrements the number of fileset entries recorded as residing on the File Server machine specified with **-server** in the FLDB entry for the server.
- If the indicated fileset uses Release Replication and the specified site is on the same File Server machine as the read/write fileset, the command removes the replica (if it exists); see the **Cautions** section for more information. For any other replica, the command instructs the Replication Server at the site to remove the replica.

Other replication sites of the read/write fileset are not affected. If the command is used to remove a fileset's last replication site, the status flag for the read-only version in the fileset's FLDB entry is set to **invalid**. If it is used to remove the last existing version of a fileset, the fileset's entire FLDB entry is removed.

Before you use the **fts delete** command to remove the read/write (and backup) version of a fileset, use the **fts rmsite** command to remove the fileset's replication sites. If Release Replication was used for the fileset, use the **fts rmsite** command to remove the replication site (and replica) stored on the same File Server machine as the read/write fileset as well.

If the aggregate on which the replication site is defined is not currently exported or has been detached with the **dfsexport** command, you must specify the aggregate ID of the aggregate; otherwise, the **fts rmsite** command cannot remove the replication site. If the aggregate is not exported or has been detached, the Replication Server on the File Server machine on which the aggregate resides stops trying to maintain the replica at the site once the **fts rmsite** command is issued, and it removes the replica from the site once the aggregate is again exported.

Replication sites are added with the **fts addsite** command. The replication type for a read/write fileset is set or changed with the **fts setrepinfo** command.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of the fileset for which the replication site and replica are to be removed. The issuer must also be listed in the **admin.ft** file on the machine specified by **-server** if the following are true:

- Release Replication is used for the fileset.
- The replication site on the same File Server machine as the read/write fileset is to be removed (in which case **-server** names the File Server machine on which the read/write fileset resides).
- A replica actually exists at the specified replication site.

CAUTIONS

If you use Release Replication and you remove the read-only fileset that is on the same File Server machine as the read/write source, all other read-only filesets become unavailable upon the expiration of the fileset's FailAge parameter. The FailAge parameter is set using the **fts setrepinfo** command.

EXAMPLES

The following command removes the replication site on the aggregate **/dev/lv01** of the File Server machine **fs5** from the FLDB entry for the fileset named **rs_aix41.bin**. A replica of **rs_aix41.bin** that resides at the site is also removed.

```
$ fts rmsite rs_aix41.bin /.../abc.com/hosts/fs5 /dev/lv01
```

RELATED INFORMATION

Commands: **fts addsite(8dfs)**, **fts delete(8dfs)**, **fts setrepinfo(8dfs)**.

Files: **dfstab(4dfs)**.

fts setprotectlevels

Purpose

Sets advisory DCE remote procedure call (RPC) authentication levels for a specified fileset.

Synopsis

fts setprotectlevels -fileset { *name* | *ID*}[**-minlocalprotectlevel** *level*][**-maxlocalprotectlevel** *level*][**-minremoteprotectlevel** *level*] [**-maxremoteprotectlevel** *level*][**-cell** *cellname*][**-verbose] [-noauth** | **-localauth] [-help]**

OPTIONS

-fileset { *name***|** *ID***}**
Specifies a fileset either by its name or volume ID.

-minlocalprotectlevel *level*
Specifies the advisory lower bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within the same cell). The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-maxlocalprotectlevel *level*
Specifies the advisory upper bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within the same cell). The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-minremoteprotectlevel *level*
Specifies the advisory lower bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within foreign cells). The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-maxremoteprotectlevel *level*
Specifies the advisory upper bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within foreign cells). The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Description section.

-cell *cellname*
Specifies the cell as *cellname* within which the specified fileset resides.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged

into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts setprotectlevels** command adjusts the minimum and maximum advisory DCE RPC authentication level bounds for a specified fileset. These bounds are used to bias a Cache Manager to a higher or lower security level when accessing the specified fileset. However, the bounds are simply advisory in that if the Cache Manager's security level settings are outside of the advisory bounds, the Cache Manager can cross the advisory and continue negotiating with a File Server. In this case, the Cache Manager's minimum security level (set with the **dfsd** or **cm setprotectlevels** command) and the File Server's maximum security bound (set with the **fxd** command) become the ″hard″ limits. Note that if the **fts setprotectlevels** bounds fall outside of File Server bounds, the File Server bounds take precedence.

In practice, when a Cache Manager must access a given fileset it first consults a Fileset Location (FL) Server for the location of that fileset (or any replicas if it is replicated read-only fileset). Along with the location, the Cache Manager also receives the applicable minimum and maximum advisory bounds for that fileset. The Cache Manager then checks its initial authentication level and compares that to the range defined by the bounds. The Cache Manager then adjusts its initial authentication level as follows:

- If the Cache Manager's initial authentication level is within the range defined by the advisory bounds, the initial level is used without adjustment.
- If the Cache Manager's initial authentication level is above the maximum advisory bound, the Cache Manager adjusts the initial level to match the advisory upper bound. However, the Cache Manager will not adjust its authentication level below its own minimum setting.
- If the Cache Manager's initial authentication level is below the minimum advisory bound, the Cache Manager adjusts the initial level to match the advisory lower bound.

The negotiation process to set an RPC authentication level now occurs as usual between the Cache Manager and File Server. The Cache Manager sends an RPC using the initial authentication level (which may have been adjusted because of the advisory bounds) to the File Server. If the initial authentication level is outside the minimum or maximum bounds set at the File Server, the File Server returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Server requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Server). Once the Cache Manager and File Server have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Server.

Note that the use of this command does not preclude communication with Cache Managers running earlier versions of DCE.

The various authentication levels are set by specifying either an integer value between 0 and 6, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **0** or **default** or **rpc_protect_level_default**: Use the DCE default authentication level.
- **1** or **none** or **rpc_protect_level_none**: Perform no authentication.
- **2** or **connect** or **rpc_protect_level_connect**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **3** or **call** or **rpc_protect_level_call**: Authenticate only at the beginning of each RPC received.
- **4** or **pkt** or **rpc_protect_level_pkt**: Ensure that all data received is from the expected host.
- **5** or **pkt_integrity** or **rpc_protect_level_pkt_integrity**: Authenticate and verify that none of the data transferred has been modified.
- **6** or **pkt_privacy** or **rpc_protect_level_pkt_privacy**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

Privilege Required

The issuer must have FLDB administration privileges or must be in the owner group for the File Server.

EXAMPLES

The following command sets the following authentication values:

- The maximum advisory authentication level for communication with Cache Managers in the local cell is set to packet integrity.
- The minimum advisory authentication level for communication with Cache Managers in the local cell is set to packet.
- The maximum advisory authentication level for communication with Cache Managers in foreign cells is set to packet security.
- The minimum advisory authentication level for communication with Cache Managers in foreign cells is set to packet security.

```
$ fts setprotectlevels -fileset richland.12 -maxlocalprotectlevel 5 -minlocalprotectlevel 4
-maxremoteprotectlevel 6 -minremoteprotectlevel 6
```

RELATED INFORMATION

Commands: **fts getprotectlevels(8dfs)**, **fxd(8dfs)**, **dfsd(8dfs)**, **fts setprotectlevels(8dfs)**

fts setquota

Purpose

Sets the maximum quota for a read/write DCE LFS fileset

Synopsis

```
fts setquota -path { filename | directory_name} | -fileset { name | ID}}
-size kbytes[-cell cellname][-noauth | -localauth [-verbose ][-help ]
```

OPTIONS

-path { *filename* | *directory_name*}
Names a directory or file located on the read/write fileset whose quota is to be set. Use this option or use **-fileset**.

-fileset { *name* | *ID*}
Specifies the complete name or fileset ID number of the read/write fileset whose quota is to be set. Use this option or use **-path**.

-size *kbytes*
Specifies the maximum amount of disk space that all of the files and directories in the read/write fileset can occupy. This includes files and directories in the read/write version of the fileset that are actually pointers to disk blocks in the backup or read-only version of the fileset. Specify the value in 1-kilobyte blocks. (A value of 1024 kilobytes is 1 megabyte.) By default, every newly created fileset has a quota of 5000 kilobytes.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts setquota** command sets the quota limit for a read/write DCE LFS fileset. (It cannot be used to set the quota for a non-LFS fileset or for a read-only or backup DCE LFS fileset.) The fileset whose quota is to be set can be indicated by

specifying the name of a file or directory in the fileset with the **-path** option or by indicating the fileset directly with the **-fileset** option.

Quota refers to the amount of disk space occupied by all of the files and directories in the read/write version of the fileset. This includes files and directories in the read/write version of the fileset that are actually pointers to disk blocks in the backup or read-only version of the fileset. Do not confuse quota with allocation; the latter identifies the amount of disk space occupied by the data that a fileset actually houses, excluding those files and directories that are pointers to disk blocks in another version of the fileset.

By default, every newly created fileset has a quota of 5000 kilobytes. This command increases or decreases a fileset's quota to the number of kilobytes specified with the **-size** option. Because it does not represent the amount of physical data the fileset contains, a fileset's quota can be larger than the size of the aggregate it resides on. Similarly, the sum of the quotas of all filesets on an aggregate can exceed the size of the aggregate.

The **fts lsft**, **fts lsheader**, and **fts lsquota** commands display, among other things, the current quota for a fileset.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine on which the fileset is stored.

EXAMPLES

The following command sets the quota for the fileset that contains the directory named */usr/terry* to 15,000 kilobytes:

```
$ fts setq /usr/terry 15000
```

RELATED INFORMATION

Commands: **fts lsft(8dfs)**, **fts lsheader(8dfs)**, **fts lsquota(8dfs)**.

fts setrepinfo

Purpose

Sets or changes replication type and parameters for a read/write DCE LFS fileset

Synopsis

```
fts setrepinfo -fileset { name | ID}[{-release | -scheduled }][-change ]
[-maxage interval][-failage interval][-reclaimwait interval]
[-minrepdelay interval][-defaultsiteage interval][-clear ][-cell  cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* | *ID*}
Specifies the complete name or fileset ID number of the read/write source fileset for which the replication type and parameters are to be set or changed. This command is used to set parameters for either Release or Scheduled Replication.

-release
Specifies that Release Replication is to be used with the fileset indicated with the **-fileset** option. When initially defining a fileset's replication parameters, use this option or use the **-scheduled** option. Afterward, omit both options when modifying the fileset's replication parameters without changing its replication type.

To change a fileset's replication type (from Release to Scheduled, or from Scheduled to Release), include both the **-change** option and either the **-release** or **-scheduled** option to indicate the new type of replication to be used with the fileset.

-scheduled
Specifies that Scheduled Replication is to be used with the fileset indicated with the **-fileset** option. When initially defining a fileset's replication parameters, use this option or use the **-release** option. Afterward, omit both options when modifying the fileset's replication parameters without changing its replication type.

To change a fileset's replication type (from Release to Scheduled, or from Scheduled to Release), include both the **-change** option and either the **-release** or **-scheduled** option to indicate the new type of replication to be used with the fileset.

-change
Specifies that the type of replication currently used with the fileset indicated with the **-fileset** option is to be changed. Include the **-release** option to change the fileset's replication type from Scheduled to Release; include the **-scheduled** option to change the fileset's replication type from Release to Scheduled.

Omit this option when specifying the **-release** or **-scheduled** option to initially set a fileset's replication type. Also omit this option when changing a fileset's replication parameters without changing its replication type.

-maxage *interval*
Specifies the amount of time the Cache Manager distributes data cached from a read-only replica without attempting to verify that the data is current. The Replication Server maintains information about the currentness of a

read-only replica, which it communicates to the Cache Manager via the File Exporter. For Scheduled Replication, a replica must remain current with respect to the read/write source fileset; for Release Replication, a replica must remain current with respect to the read-only fileset that resides on the same File Server machine as the read/write source fileset. The default is 2 hours. An effective value must be greater than or equal to 2 minutes. *Applicable to Release and Scheduled Replication.*

-failage *interval*
Specifies the amount of time the Cache Manager distributes data cached from a read-only replica if that data cannot be verified as current. The difference between FailAge and MaxAge is the amount of time the Cache Manager continues to distribute data cached from a read-only replica after that data cannot be verified as current. The default is 1 day or twice the MaxAge, whichever is larger. An effective value must be greater than or equal to the MaxAge. *Applicable to Release and Scheduled Replication.*

-reclaimwait *interval*
Specifies the amount of time the File Exporter waits before it reclaims storage space from deleted files—those not referred to by a directory (ReclaimWait). It also determines the frequency of the Cache Manager's keep-alive messages to the Replication Server.

The Cache Manager sends keep-alive messages to indicate that it is still using files on a read-only replica. A file being accessed from a replica remains available as long as the Cache Manager continues to notify the Replication Server that the file is still in use and the Replication Server continues to forward these notifications to the File Exporter. This is true even if the file has been removed from all directories on the read/write fileset in the interim. To prevent the File Exporter from reclaiming storage space occupied by deleted files, the Cache Manager sends keep-alive messages more frequently than the ReclaimWait interval. The default is 18 hours. An effective value must be greater than 2 hours; do not specify a value less than 90 minutes. *Applicable to Release and Scheduled Replication.*

-minrepdelay *interval*
Specifies how long the Replication Server waits after a read/write source fileset changes before it attempts to get a new copy of the fileset (MinRepDelay). The Replication Server tracks the currentness of replicas by maintaining a whole-fileset token for each fileset. If a Cache Manager changes the read/write fileset, the Replication Server relinquishes its whole-fileset token and waits for at least the time specified by MinRepDelay before requesting a new whole-fileset token. The default is 5 minutes or one quarter of the DefaultSiteAge, whichever is smaller. This value must be less than the MaxSiteAge specified for each replication site with the **-maxsiteage** option of the **fts addsite** command. *Applicable to Scheduled Replication only.*

-defaultsiteage *interval*
Specifies the default value to be used as the MaxSiteAge for a replication site (DefaultSiteAge). The DefaultSiteAge is used if the **-maxsiteage** option is omitted when the **fts addsite** command is used to add a replication site. The default is one quarter of the MaxAge. *Applicable to Scheduled Replication only.*

-clear Removes all replication parameters previously defined for the fileset. The options associated with the type of replication in use for the fileset can then

be used to define new replication parameters, or they can all be omitted to allow the system to calculate new replication parameters for the fileset.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts setrepinfo** command is used to set or change the replication type and parameters for a read/write DCE LFS fileset. It affects the parameters for both Release and Scheduled Replication. It must be issued before replication sites can be defined for the fileset with the **fts addsite** command and before the **fts release** or **fts update** command can be used to copy replicas to the replication sites. The replication type and parameters for a fileset are stored in the fileset's entry in the Fileset Location Database (FLDB).

Use the following guidelines when deciding which type of replication (Release or Scheduled) to use with a read/write fileset:

- Use Release Replication if the fileset seldom changes or if the distribution of replicas must be tracked closely.
- Use Scheduled Replication if having the system release replicas of the fileset at regular intervals is preferred and the distribution of replicas does not need to be tracked.

When initially defining a fileset's replication type, include either the -release or **-scheduled** option. These options are then omitted from the command unless the replication type for the fileset is being changed (from Release to Scheduled, or from Scheduled to Release). To change the replication type, use the appropriate option (-release or **-scheduled**) to specify the new type, and include the **-change** option to indicate that the type is to be changed.

Note that, because Release Replication does not require a replication site to have a MaxSiteAge, it is likely that one or more Release Replication sites will have a MaxSiteAge of **0** (zero), which is the default value recorded for a site if no MaxSiteAge or DefaultSiteAge is specified. When changing from Release Replication to Scheduled Replication, the **-defaultsiteage** option *must* be used to set a DefaultSiteAge if any replication site does not have a MaxSiteAge and no

DefaultSiteAge exists for the source fileset; otherwise, the **fts setrepinfo** command fails. If the command fails for this reason, reissue it, specifying a DefaultSiteAge with the **-defaultsiteage** option.

The **-maxage**, **-failage**, **-reclaimwait**, **-minrepdelay**, and **-defaultsiteage** options are used to set the corresponding replication parameters for a read/write fileset. (See the section on options for information on the replication parameter each option sets.) The following table lists each option's default value and describes the dependencies between the different options when they are used to set the replication parameters for either Release or Scheduled Replication.

Parameter	Default	Release Replication	Scheduled Replication
-maxage	2 hours	*Required only if* **-failage** is specified.	*Required only if* one of the following is specified: **-failage** , **-minrepdelay** , or **-defaultsiteage**.
-failage	The larger of 1 day or twice **-maxage**	*Optional.* If it is specified, the following are required: **-maxage** and **-reclaimwait**.	*Required only if* one of the following is specified: **-minrepdelay** or **-defaultsiteage**.
-reclaimwait	18 hours	*Required only if* **-failage** is specified.	*Required only if* one of the following is specified: **-failage** , **-minrepdelay** , or **-defaultsiteage**.
-minrepdelay	The smaller of 5 minutes or one quarter of **-defaultsiteage**	*Not applicable.*	*Required only if* one of the following is specified: **-failage** or **-defaultsiteage**.
-defaultsiteage	One-quarter of **-maxage**	*Not applicable.*	*Optional.* But if the other options are specified and **-defaultsiteage** is not, the **-maxsiteage** option of the `fts addsite` command is required when defining replication sites for the fileset.

The following options are used to define the parameters for Release Replication (the **-minrepdelay** and **-defaultsiteage** options do not apply for Release Replication):

- **-maxage** is required only if **-failage** is specified. Otherwise, the default is 2 hours.

- **-failage** is optional. If it is specified, both **-maxage** and **-reclaimwait** are required. The default is 1 day or twice the MaxAge, whichever is longer.
- **-reclaimwait**is required only if **-failage** is specified. Otherwise, the default is 18 hours.

The following options are used to define the parameters for Scheduled Replication:

- **-maxage** is required only if **-failage**,**-minrepdelay**, or **-defaultsiteage** is specified. Otherwise, the default is 2 hours.
- **-failage** is required only if **-minrepdelay** or **-defaultsiteage** is specified. Otherwise, the defualt is 1 day or twice the MaxAge, whichever is larger.
- **-reclaimwait** is required only if **-failage**, **-minrepdelay**, or **-defaultsiteage** is specified. Otherwise, the default is 18 hours.
- **-minrepdelay** is required only if **-failage** or **-defaultsiteage** is specified. Otherwise, the default is 5 minutes or one-quarter of the DefaultSiteAge, whichever is smaller.
- **-defaultsiteage** is always optional. The defualt is one-quarter of the MaxAge. However, if the other four options are specified and **-defaultsiteage** is not, the **-maxsiteage** option must be specified when defining replication sites for the read-write fileset with the **fts addsite** command.

The **fts** program calculates default values for each of the parameters *unless*

- The **-failage** option is specified for Release Replication.
- The **-failage**, **-minrepdelay**, or **-defaultsiteage** option is specified for Scheduled Replication.

Once one of these options is specified, the **fts** program no longer performs any default calculations; *interval* must be provided for all applicable options. (The exception is the **-defaultsiteage** option for Scheduled Replication, which is always optional.) Also, because the **-minrepdelay** and **-defaultsiteage** options do not apply to Release Replication, they are recorded if specified but they are ignored.

Enter *interval* values as integers, using the following abbreviations to indicate units: **d** for days, **h** for hours, **m** for minutes, and **s** for seconds. The syntax for an *interval* is

[*integer***d**] [*integer***h**] [*integer***m**] [*integer***s**]

At least one of the four values (days, hours, minutes, or seconds) must be provided, and a unit abbreviation (**d**, **h**, **m**, or **s**) must be used with any integer. The unit abbreviations can be uppercase or lowercase, and they can be specified in any order. Examples of valid *interval* values are

```
3d2H
3M2h
1d6h30m45s
```

To change the replication parameters defined for a fileset, use the options for the parameters you want to change. To change *all* replication parameters associated with a fileset, use the **-clear** option to remove all replication parameters previously defined for the fileset, and use the options for the parameters you want to change to indicate the new parameters. To have the system calculate default values for all replication parameters, use only the **-clear** option.

Use the **fts lsfldb** or **fts lsft** command to display the replication parameters associated with a read/write fileset. Use the **fts lsreplicas** command to display the

statuses of replicas at replication sites. Use the **fts statrepserver** command to display the status of the Replication Server on a File Server machine.

Note that replication is available in a cell only if the following conditions have been met: **root.dfs**, the cell's main read/write fileset, is a DCE LFS fileset; **root.dfs** was mounted with an explicit read/write mount point as a subdirectory of itself (the **root.dfs** fileset) when the cell was configured; and **root.dfs** is replicated. See Part 1 of this manual for information about configuring **root.dfs** to support replication.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset resides. The issuer must also be listed in the **admin.ft** file on the machine on which the read/write fileset resides if the following are true: The fileset's replication type is being changed from Release Replication to Scheduled Replication, and a replica actually resides at the replication site on the same File Server machine as the read/write fileset. (The first replication site defined for a fileset that uses Release Replication must be on the same File Server machine as the read/write fileset.)

CAUTIONS

When using the **fts setrepinfo** command to set replication parameters, it is recommended that the default parameters (with the exception of MaxAge) be used for both types of replication. The dependencies between the parameters are complicated and should be defined by the issuer only when absolutely necessary.

RELATED INFORMATION

Commands: **fts addsite(8dfs)**, **fts lsfldb(8dfs)**, **fts lsft(8dfs)**, **fts lsreplicas(8dfs)**, **fts release(8dfs)**, **fts rmsite(8dfs)**, **fts statrepserver(8dfs)**, **fts update(8dfs)**.

fts statftserver

Purpose

Reports on the activity of a Fileset Server

Synopsis

fts statftserver-server *machine* **-cell** *cellname*][**-noauth** | **-localauth**]
[**-verbose][-help**]

OPTIONS

-server *machine*
Names the File Server machine about whose Fileset Server information is to be reported. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts statftserver** command reports on the actions of the Fileset Server (**ftserver** process) on the File Server machine specified with the **-server** option. The command returns information about the actions of the Fileset Server at the moment it is issued. This command is useful mainly if there is concern that a Fileset Server is not performing requested actions.

If no transactions are active on the specified machine, the command displays a message to that effect. This indicates that the Fileset Server is functioning properly. If transactions are active on the machine, the command displays information about the action currently being performed by the Fileset Server. Depending on the information displayed, the Fileset Server may or may not be functioning properly.

OUTPUT

If the Fileset Server is not currently performing any actions, the command displays the following message, indicating that the Fileset Server is functioning normally:

```
No active transactions on machine_name
```

If the Fileset Server is currently performing an action, the command displays information about the actions of the Fileset Server. The output includes fields containing ID numbers and flags that the Fileset Server sets for internal use. The details of the information returned by the command are more useful to programmers than to system administrators. A full understanding of the output requires familiarity with the code for the Fileset Server.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine specified by **-server**.

RELATED INFORMATION

Commands: **ftserver(8dfs)**.

fts statrepserver

Purpose

Displays the status of a Replication Server

Synopsis

fts statrepserver -server *machine*[**-long**][**-cell** *cellname*][**-noauth** | **-localauth** [**-verbose**][**-help**]

OPTIONS

-server *machine*
Names the File Server machine about whose Replication Server status information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-long Specifies that more detailed information about the Replication Server is to be displayed. The additional output includes information about each replica managed by the Replication Server on the specified machine.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other options specified with this option are ignored.

DESCRIPTION

The **fts statrepserver** command displays information about the status of the Replication Server (**repserver** process) on the File Server machine specified with the **-server** option. Include the **-long** option to specify more detailed information about the Replication Server on the specified machine, as well as information about each replica managed by the Replication Server. Use the **fts lsreplicas** command to check the status of each replica of a fileset.

Privilege Required

No privileges are required.

RELATED INFORMATION

Commands: **fts lsreplicas(8dfs)**, **repserver(8dfs)**.

fts syncfldb

Purpose

Synchronizes FLDB entries to match their fileset headers

Synopsis

```
fts syncfldb -server machine[-aggregate name][-cell cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-server *machine*
Names the File Server machine from which to compare filesets to entries in the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** for which to compare filesets to FLDB entries. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file. Do not use this option under normal circumstances; omitting it allows synchronization of all filesets on **-server**. Use it only when just a single aggregate needs to be synchronized.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts syncfldb** command inspects the fileset header of each online fileset that resides on a specified File Server machine (or, optionally, a specified aggregate or partition on that File Server machine). The command then checks that each FLDB entry is consistent with its fileset header. If the command encounters an

inconsistency between a fileset header and its FLDB entry, the FLDB entry is changed to reflect the information in the fileset header. If the command encounters an FLDB entry without a corresponding fileset header, it deletes the FLDB entry; if the command encounters a fileset header without a corresponding FLDB entry, it creates an FLDB entry for that fileset.

Note: The **fts syncfldb** command synchronizes only the FLDB entries for filesets on aggregates that are exported from the DFS File Server machine at the time the command is run. If fileset entries are found in the FLDB but the aggregate is not exported at that time, no action is taken.

The **fts syncfldb** command also performs the following additional functions:

- If it finds a backup fileset whose read/write source no longer exists at the same site, it displays a warning message.
- If it finds a fileset ID number that is larger than the value of the counter used by the FL Server when allocating fileset ID numbers, it records this ID number as the new value of the counter. The next fileset to be created receives a fileset ID number one greater than this number.
- If necessary, it increments or decrements the number of fileset entries recorded as residing on a File Server machine in the FLDB entry for the server.

The **fts syncfldb** command checks either all of the fileset headers on the File Server machine specified with the **-server** option or only the filesets on the optional partition or aggregate specified with the **-aggregate** option. The command checks a fileset header only if the fileset is marked as being **On-line**. If the command encounters a busy fileset on an aggregate, it exits without checking any other filesets. (A busy fileset is one upon which a fileset-related operation such as a move, clone, or release is currently being performed.)

It is recommended that the **fts syncfldb** command be run on all File Server machines in a cell *before* the **fts syncserv** command is run on the File Server machines in the cell. However, nothing prohibits the commands from being executed in the reverse order or independently of each other.

Note that the **fts syncfldb** and **fts syncserv** commands cannot restore replication information lost when the entry for a DCE LFS fileset is removed from the FLDB. Replication information must be reconstructed with the **fts setrepinfo** and **fts addsite** commands.

Because non-LFS filesets do not have fileset headers, the **fts syncfldb** and **fts syncserv** commands have limited effectiveness on non-LFS filesets. For example, because non-LFS filesets do not have fileset headers, the **fts syncfldb** command cannot determine the name of a non-LFS fileset that has no FLDB entry. If the command determines that it needs to create an FLDB entry for a non-LFS fileset, it generates a name of the form **SYNCFLDB-ADDED-** *number*, where *number* is a unique number appended to the name to differentiate it from other names of the same type. The **fts rename** command then needs to be used to rename the fileset to its original name.

Note: If an entry exists in the fileset but not in the FLDB, no action is taken.

Privilege Required

The issuer must be listed in the **admin.ft** file on each machine that houses a version of any fileset stored at the specified site (-server and optionally **-aggregate**

). The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset stored at the specified site.

CAUTIONS

The physical disk on which a fileset resides cannot be moved from a machine in one cell to a machine in another cell with the expectation of simply running the **fts syncfldb** command to create an FLDB entry for the fileset in the new cell. Any attempt to introduce a fileset from one cell into another cell risks a fileset ID conflict between the newly introduced fileset and a fileset within the new cell that has the same fileset ID. This conflict causes one of the two conflicting filesets to be inaccessible.

RELATED INFORMATION

Commands: **fts addsite(8dfs)**, **fts rename**, **fts setrepinfo(8dfs)**, **fts syncserv(8dfs)**.

Files: **dfstab(4dfs)**.

fts syncserv

Purpose

Synchronizes fileset headers to match their FLDB entries

Synopsis

fts syncserv -server *machine*[**-aggregate** *name*][**-cell** *cellname*]
[**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-server *machine*
Names the File Server machine for which to check entries in the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** for which to check FLDB entries. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the */dce/local/***var/dfs/dfstab** file. Do not use this option under normal circumstances; omitting it allows synchronization of all filesets on **-server**. Use it only when just a single aggregate needs to be synchronized.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts syncserv** command inspects the FLDB entry of each fileset on a specified File Server machine (or, optionally, a specified aggregate or partition on that File Server machine). The command then checks that each fileset header is consistent with its FLDB entry. If the command finds an inconsistency between the fileset

name found in the fileset header and the name found in the FLDB entry, the fileset header is renamed to reflect the name in the FLDB entry. If the command encounters a fileset marked as **Off-line**, but the fileset's FLDB entry lists it as being **valid**, the command places the fileset online.

The **fts syncserv** command checks either all of the filesets on the File Server machine specified with the **-server** option or only the filesets on the optionally specified partition or aggregate specified with the **-aggregate** option. The command also checks the reported sites of all copies of an inspected fileset (even though that requires checking filesets on server machines other than **-server**).

It is recommended that the **fts syncfldb** command be run on all File Server machines in a cell *before* the **fts syncserv** command is run on the File Server machines in the cell. However, nothing prohibits the commands from being executed in the reverse order or independently of each other.

Note that the **fts syncserv** and **fts syncfldb** commands cannot restore replication information lost when the entry for a DCE LFS fileset is removed from the FLDB. Replication information must be reconstructed with the **fts setrepinfo** and **fts addsite** commands.

Because non-LFS filesets do not have fileset headers, the **fts syncserv** and **fts syncfldb** commands have limited effectiveness on non-LFS filesets. For example, because the **fts syncserv** command cannot destroy a disk partition, it cannot delete a non-LFS fileset, even if it determines that the fileset needs to be deleted. Instead, the **fts** program displays a warning message reporting the non-LFS fileset that needs to be deleted to restore file system consistency. The proper commands then need to be used to delete the fileset.

Privilege Required

The issuer must be listed in the **admin.ft** file on each machine that houses a version of any fileset stored at the specified site (-server and optionally **-aggregate**). The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset stored at the specified site.

EXAMPLES

The following command synchronizes the FLDB entries of filesets whose site definitions mention **fs3**, including any copies of the filesets not located on **fs3**:

```
$ fts syncserv /.../abc.com/hosts/fs3
```

RELATED INFORMATION

Commands: **fts addsite(8dfs)**, **fts setrepinfo(8dfs)**, **fts syncfldb(8dfs)**.

Files: **dfstab(4dfs)**.

fts unlock

Purpose

Unlocks an entry in the FLDB

Synopsis

fts unlock -fileset { *name* | *ID*}[**-cell** *cellname*][**-noauth** | **-localauth]**
[-verbose][-help]

OPTIONS

-fileset { *name* **|** *ID***}**
Specifies the complete name or fileset ID number of the fileset whose entry in the Fileset Location Database (FLDB) is to be unlocked.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts unlock** command releases the lock on the FLDB entry for the fileset indicated by **-fileset**. Use the **fts unlockfldb** command to unlock multiple filesets at one time.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be unlocked resides.

CAUTIONS

Do not issue this command under normal circumstances. It is useful only if the FLDB entry for a fileset is locked but there is no reason to suspect inconsistency

within the fileset or between it and the FLDB. Note that it is possible to list information from locked FLDB entries, even though they cannot be manipulated in other ways.

EXAMPLES

The following command unlocks the FLDB entry for the fileset named *user.terry*:

```
$ fts unlock user.terry
```

RELATED INFORMATION

Commands: **fts lock(8dfs)**, **fts unlockfldb(8dfs)**.

fts unlockfldb

Purpose

Unlocks all specified locked entries in the FLDB

Synopsis

fts unlockfldb -server *machine*][**-aggregate** *name*][**-cell** *cellname*]
[**-noauth** | **-localauth][-verbose][-help]**

OPTIONS

-server *machine*
: Names the File Server machine whose filesets are to have their Fileset Location Database (FLDB) entries unlocked. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option with **-aggregate** to unlock the entries for the filesets on a specific aggregate on **-server**. Omit both this option and **-aggregate** to unlock all of the entries in the FLDB.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of an aggregate or partition on **-server** on which the filesets whose FLDB entries are to be unlocked reside. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal***/var/dfs/dfstab** file. The **-server** option must be specified with this option. Omit both this option and **-server** to unlock all of the entries in the FLDB.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts unlockfldb** command releases the locks on the FLDB entries indicated by the combination of options specified. To unlock

- All entries in the FLDB, specify no options.
- All entries that mention a File Server machine in a site definition, specify the name of the File Server machine with **-server**.
- All entries that mention a specific site, specify both **-server** and **-aggregate**.
- A single fileset, use the **fts unlock** command instead.

Privilege Required

The issuer must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset to be unlocked.

CAUTIONS

Do not issue this command under normal circumstances. It is useful only if FLDB entries for filesets at a certain site are locked, but there is no reason to suspect inconsistency within the filesets or between the filesets and the FLDB. Note that it is possible to list information from locked FLDB entries, even though they cannot be manipulated in other ways.

EXAMPLES

The following command unlocks all locked entries in the FLDB:

```
$ fts unlockfldb
```

RELATED INFORMATION

Commands: **fts lock(8dfs)**, **fts unlock(8dfs)**.

Files: **dfstab(4dfs)**.

fts update

Purpose

Requests an immediate update of replicas of a read/write DCE LFS fileset that uses Scheduled Replication

Synopsis

```
fts update -fileset {name | ID}{-all | -server machine}[-cell cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-fileset { *name* | *ID*}
Specifies the complete name or fileset ID number of the read/write fileset whose replicas are to be updated immediately. For a fileset that uses Scheduled Replication, the command updates the indicated replicas to match the read/write version of the fileset. For a fileset that uses Release Replication, the command updates the replicas to match the read-only version that resides at the same site as the read/write version of the fileset.

-all Specifies that all replicas of the fileset indicated with the **-fileset** option are to be updated. Use this option or use the **-server** option.

-server *machine*
Names a specific File Server machine on which the replica of the fileset indicated with the **-fileset** option is to be updated. Only the replica on the specified File Server machine is updated. Specify the machine's DCE pathname, its host name, or its IP address. Use this option or use the **-all** option.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts update** command asks the Replication Server to make an immediate update of replicas of the read/write DCE LFS fileset specified with the **-fileset** option. The effect of the command depends on whether the fileset to be updated uses Scheduled or Release Replication, as follows:

- *For a fileset that uses Scheduled Replication*, the command directs the Replication Servers on the indicated File Server machines to perform an immediate update based on the read/write version of the fileset. The Replication Servers ignore the value of the MinRepDelay parameter associated with the fileset; they immediately begin updating the replicas to match the version of the read/write fileset that exists at the time the command is issued.
- *For a fileset that uses Release Replication*, the command directs the Replication Servers on the indicated File Server machines to perform an immediate update based on the read-only replica that is stored on the same File Server machine as the read/write fileset (the replica that was created when the **fts release** command was last used for the fileset). The command does *not* first update the read-only replica on the read/write fileset's File Server machine. Because the MinRepDelay parameter does not apply to a fileset that uses Release Replication, the replicas should already be updated to match the replica on the read/write fileset's machine; the command should have no effect.

To indicate the replicas of the specified fileset that are to be updated, use the command's **-all** or **-server** option as follows:

- To update all replicas of the specified fileset, use the **-all** option.
- To update the replica stored on a specific File Server machine, identity the machine with the **-server** option.

Note that, as with releasing a new version of a fileset that uses Release Replication, forcing an update of a fileset that uses Scheduled Replication does not ensure immediate access to data in the new version of the replica. A Cache Manager continues to provide data cached from the old version of the replica until the MaxAge for the fileset expires or until the Cache Manager needs to access data from the replica that it has not already cached.

To gain immediate access to data in the new version of the replica, issue the **cm flush** or **cm flushfileset** command to flush the old data from the cache. This forces the Cache Manager to replace data it has cached from the replica. Replication Servers begin replication in parallel; however, until all replicas have been updated, you cannot directly force the Cache Manager to access data from the new version of the replica.

The **fts update** command does not change the replication type and parameters defined for the specified fileset. Before the **fts update** command can be used, the **fts setrepinfo** command must be used to define the replication parameters for the read/write fileset. The **fts addsite** command must also be used to define at least one replication site for the read/write fileset.

Use the **fts lsreplicas** command to check the status of replicas of the fileset. Use the **fts statrepserver** command to check the status of the Replication Server on a File Server machine.

Privilege Required

No privileges are required.

EXAMPLES

The following command requests an immediate update of the replica of the read/write fileset named **sparc_sunos53.bin** at the replication site defined on the File Server machine named **fs3**:

```
$ fts update sparc_sunos53.bin /.../abc.com/hosts/fs3
```

RELATED INFORMATION

Commands: **cm flush(8dfs)**, **cm flushfileset(8dfs)**, **fts addsite(8dfs)**, **fts lsreplicas(8dfs)**, **fts release(8dfs)**, **fts setrepinfo(8dfs)**, **fts statrepserver(8dfs).**

fts zap

Purpose

Removes a DCE LFS fileset from its site without updating the FLDB

Synopsis

```
fts zap -ftid ID -server machine -aggregate name[-cell cellname]
[-noauth | -localauth ][-verbose ][-help ]
```

OPTIONS

-ftid *ID*
: Specifies the fileset ID number of the fileset to remove; a fileset name is not a valid argument.

-server *machine*
: Names the File Server machine from which to remove the fileset. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate on **-server** from which to remove the fileset. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the *dcelocal***/var/dfs/dfstab** file.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **fts zap** command removes the DCE LFS fileset with the fileset ID number specified by **-ftid** from the site defined by **-server** and **-aggregate**. It neither changes the corresponding Fileset Location Database (FLDB) entry for the fileset nor decrements the number of fileset entries recorded in the server entry in the FLDB for the specified File Server machine.

This command is useful in situations in which it is important to delete a fileset but, for some reason, the FLDB is unreachable (for example, the FL Server is unavailable). The issuer can remove information about the deleted fileset from the FLDB by running the **fts syncserv** and **fts syncfldb** commands. The issuer can also reconcile the FLDB with the **fts rmsite** command (which removes site information about a read-only version from a fileset's FLDB entry), the **fts delete** command (which removes site information about the read/write or backup version from a fileset's FLDB entry), or the **fts delfldbentry** command (which removes the entire entry for a fileset from the FLDB). (The `fts zap` command can also be used to remove normally temporary ″clone″ filesets that can sometimes be left after an interrupted `fts move` operation.)

If the DCE LFS fileset to be removed is also mounted locally (as a file system on its File Server machine), you must remove its local mount point before you delete it. The **fts zap** command cannot be used to delete a fileset that is mounted locally.

Privilege Required

The issuer must be listed in the **admin.ft** file on the machine specified by **-server**.

CAUTIONS

Do not use this command as the standard way to remove a fileset. It can make the FLDB inconsistent with the filesets on File Server machines. Use the **fts delete** command to remove the fileset entry from the FLDB at the same time that the fileset is removed.

EXAMPLES

The following command removes the fileset with fileset ID **0,,36870988** from **/dev/lv01** on **fs6**, without recording the change in the FLDB:

```
$ fts zap 0,,36870988 /.../abc.com/hosts/fs6 /dev/lv01
```

RELATED INFORMATION

Commands: **fts delete(8dfs)**, **fts delfldbentry(8dfs)**, **fts rmsite(8dfs)**, **fts syncfldb(8dfs)**, **fts syncserv(8dfs)**.

Files: **dfstab(4dfs)**.

ftserver

Purpose

Initializes the Fileset Server

Synopsis

ftserver[**-adminlist** *filename*][**-verbose][-help]**

OPTIONS

-adminlist *filename*
: Specifies the administrative list file that contains principals and groups authorized to execute **ftserver** RPCs (usually using **fts** commands). If this option is omitted, **ftserver** obtains the list of authorized users from the default administrative list file, *dcelocal***/var/dfs/admin.ft**.

-verbose
: Directs the **ftserver** process to provide a detailed report on what it is doing by displaying messages on standard error.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **ftserver** command. See the **bos help** and **bos apropos** reference pages for examples of using these commands.

DESCRIPTION

The Fileset Server, or **ftserver** process, handles fileset administration operations, such as creating, deleting, moving, and replicating filesets. The **ftserver** process must be run on all machines that export data for use in the global namespace. A machine that runs the Fileset Server, the File Exporter (which is initialized by the **fxd** process), and the **dfsbind** process is considered a DFS File Server machine. The Fileset Server is usually started and controlled by the BOS Server; if it is not, execute the **ftserver** process as a background process. The binary file for the **ftserver** process resides in *dcelocal***/bin/ftserver**.

The first time it is initialized, **ftserver** creates the *dcelocal***/var/dfs/admin.ft** administrative list file if the file does not already exist. The principals and groups listed in the **admin.ft** administrative list are authorized to administer filesets on the machine. Because some operations, such as fileset moves, are accomplished by two Fileset Servers communicating, server principal names must also appear in the **admin.ft** list.

For simplified administration, create one **admin.ft** administrative list that contains the server principal names of all machines in the administrative domain. The same **admin.ft** list can then be used by all **ftserver** processes in the domain.

When it is started, **ftserver** creates the *dcelocal***/var/dfs/adm/FtLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when the **ftserver** is started, the process moves it to the **FtLog.old** file in the same directory (overwriting the current **FtLog.old** file if it exists) before creating a new version to append messages to.

Use the **fts statftserver** command to check the status of the Fileset Server on any server machine.

Privilege Required

You must be logged in as **root** on the local machine.

OUTPUT

If problems are encountered during initialization, the **ftserver** process displays error messages on standard error output. The **ftserver** process keeps an event log in the file *dcelocal***/var/dfs/adm/FtLog**.

If run with the **-verbose** option, the **ftserver** process provides a detailed report on what it is doing by displaying messages on standard error.

RELATED INFORMATION

Commands: **dfsbind(8dfs)**, **fts statftserver(8dfs)**, **fxd(8dfs)**.

Files: **admin.ft(4dfs)**, **FtLog(4dfs)**.

fxd

Purpose

Initializes the File Exporter and starts associated kernel daemons

Synopsis

fxd -admingroup *group*[**-mainprocs** *number_of_background_daemons*]
[**-tokenprocs** *number_of_token_daemons*][**-hostlife** *client_timeout*]
[**-hostrpc** *client_rpc_timeout*][**-pollinterval** *server_poll_period*]
[**-maxlife** *max_hostlife*][**-maxrpc** *max_hostrpc*][**-notsr**]
[**-minlocalprotectlevel** *level*][**-maxlocalprotectlevel** *level*]
[**-minremoteprotectlevel** *level*][**-maxremoteprotectlevel** *level*]
[-verbose][-help]

OPTIONS

-admingroup *group*

Specifies the name of the group that can administer the File Exporter on this machine. Members of the specified group can effectively change the permissions, owner, and owning group of any file system object exported from the machine. A group from the local cell can be specified by a full or abbreviated group name (for example, ***/.../*** *cellname****/*** *group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name. The **-admingroup** option performs a function similar to that of the administrative lists associated with DFS server processes, such as the Fileset Server and the Fileset Location Server, that run in the user space.

-mainprocs *number_of_background_daemons*

Specifies the number of main kernel processes (File Exporter kernel daemons) to run on the machine. These File Exporter kernel daemons are responsible for receiving and servicing RPC requests for data and tokens from DFS clients. Specify an integer greater than 0 (zero) to indicate the number of main kernel daemons to perform these services. If this option is omitted, four main kernel daemons perform these services.

-tokenprocs *number_of_token_daemons*

Specifies the number of token-revocation kernel processes (File Exporter kernel daemons) to run on the machine. These File Exporter kernel daemons are responsible for responding to RPCs from DFS clients that are themselves responding to token revocation requests. Specify an integer greater than 0 (zero) to indicate the number of kernel daemons to perform these services. If this option is omitted, two kernel daemons perform these services.

-hostlife *client_timeout*

Specifies the host lifetime the File Exporter assigns to each client that contacts it. The host lifetime is the length of time for which the File Exporter considers a client to be alive. Each client must contact the File Exporter within this amount of time to renew its host lifetime. As long as a client's host lifetime has not expired, the File Exporter cannot revoke its tokens without its permission.

By default, the File Exporter assigns each client a host lifetime of 2 minutes. Specify an integer to indicate a number of seconds to serve as the host lifetime. The host lifetime must be greater than 0 (zero) and less than or equal to the maximum host lifetime (specified with the **-maxlife** option) and the host RPC lifetime (specified with the **-hostrpc** option).

-hostrpc *client_rpc_timeout*

Specifies the host RPC lifetime the File Exporter assigns to each client that contacts it. The host RPC lifetime is the length of time for which the File Exporter guarantees to attempt an RPC to a client before it revokes its tokens. The File Exporter can revoke the tokens of any client whose host RPC lifetime has expired without contacting the client.

By default, the File Exporter assigns each client a host RPC lifetime of 2 minutes. Specify an integer to indicate a number of seconds to serve as the host RPC lifetime. The host RPC lifetime must be greater than or equal to the host lifetime (specified with the **-hostlife** option) and less than or equal to the maximum host RPC lifetime (specified with the **-maxrpc** option).

-pollinterval *server_poll_period*

Specifies the polling interval the File Exporter assigns to each client that contacts it. The polling interval is the frequency with which each client that has tokens from the File Exporter is to poll it in the event that it cannot be reached. Each client sends an RPC to the File Exporter with this frequency until it can again contact it.

By default, the File Exporter assigns each client a polling interval of 3 minutes. Specify an integer greater than 0 (zero) to indicate a number of seconds to serve as the polling interval.

-maxlife *max_hostlife*

Specifies the maximum host lifetime the File Exporter can grant a client. A client can request a host lifetime larger than that specified with the **-hostlife** option, but the File Exporter will not grant a host lifetime greater than the value specified with this option.

By default, the File Exporter uses a value of 2 minutes as the maximum host lifetime. Specify an integer to indicate a number of seconds to serve as the maximum host lifetime. The maximum host lifetime must be greater than or equal to the host lifetime (specified with the **-hostlife** option) and less than or equal to the maximum host RPC lifetime (specified with the **-maxrpc** option).

-maxrpc *max_hostrpc*

Specifies the maximum host RPC lifetime the File Exporter can grant a client. A client can ask for a host RPC lifetime larger than that specified with the **-hostrpc** option, but the File Exporter will not grant a host RPC lifetime greater than the value specified with this option.

By default, the File Exporter uses a value of 2 minutes as the maximum host RPC lifetime. Specify an integer to indicate a number of seconds to serve as the maximum host RPC lifetime. The maximum host RPC lifetime must be greater than or equal to the host RPC lifetime (specified with the **-hostrpc** option) and the maximum host lifetime (specified with the **-maxlife** option).

-notsr Specifies that the File Exporter is to forego token state recovery when it is restarted. If this option is specified, the File Exporter accepts requests for new tokens as soon as it can again be contacted by clients. By default, it provides a brief token state recovery period during which it accepts requests only to reestablish tokens from clients that held them before it was restarted. (It bases the duration of its period of token state recovery on the greater of its **-pollinterval** or **-maxlife**, plus 20 seconds.)

This option is useful primarily for debugging purposes. Use it sparingly, as it prevents the File Exporter from maintaining consistent token state across File Server machine restarts.

-minlocalprotectlevel *level*
Specifies the minimum acceptable DCE RPC authentication level for communications between the File Exporter and clients within the same cell. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Security subsection in the Description section.

-maxlocalprotectlevel *level*
Specifies the maximum acceptable DCE RPC authentication level for communications between the File Exporter and clients in the local cell. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Security subsection in the Description section.

-minremoteprotectlevel *level*
Specifies the minimum acceptable DCE RPC authentication level for communications between the File Exporter and clients in foreign cells. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Security subsection in the Description section.

-maxremoteprotectlevel *level*
Specifies the maximum acceptable DCE RPC authentication level for communications between the File Exporter and clients in foreign cells. The *level* is set either as an integer value between 0 and 6, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see the Security subsection in the Description section.

-verbose
Directs **fxd** to produce more detailed information about its actions during initialization and as it creates kernel daemons.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **fxd** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The **fxd** command initializes the File Exporter on a File Server machine and starts all kernel daemons, such as those for garbage collection, required by the File Exporter. During initialization, it also passes the File Exporter such information as the name of the local cell, information about the local Fileset Database machines, and the identity of the group that can administer the File Exporter. The File Exporter uses this information to communicate with other processes, such as the Fileset Location (FL) Servers on Fileset Database machines, and to ensure that only privileged users administer data in filesets exported from the machine.

The File Exporter must be run on all machines that export data for use in the global namespace. A machine that runs the File Exporter, the Fileset Server (**ftserver** process), and the **dfsbind** process is considered to be a DFS File Server machine. The File Exporter is typically run by adding the **fxd** command to the proper start-up file (**/etc/rc** or its equivalent). The **dfsbind** process must be run before the **fxd** process in a start-up file. The binary file for the **fxd** process resides in */dcelocal/***bin/fxd**. The process automatically places itself in the background once its initialization is complete.

The **-mainprocs** and **-tokenprocs** options can be used to alter the default number of kernel daemons running on the server machine, as follows:

- The **-mainprocs** option specifies the number of main kernel daemons that run on the machine to service RPC requests for data and tokens from DFS clients. The default number of main kernel daemons is four, which is usually sufficient to handle RPC requests from many DFS client machines. Use the **-mainprocs** option to increase the number of main kernel daemons dedicated to servicing RPC requests if the machine is to support a large number of DFS clients.
- The **-tokenprocs** option specifies the number of kernel daemons dedicated to responding to RPCs from DFS clients that are themselves responding to token revocation requests from the File Exporter. The default number of kernel daemons dedicated to this task is two. If the **-mainprocs** option is used to increase the number of main kernel daemons, use the **-tokenprocs** option to increase the number of kernel daemons dedicated to handling responses to token revocation requests accordingly.

On most system types, these kernel daemons appear as nameless entries in the output of the **ps** command (or its equivalent). However, because some of the kernel daemons run as threads rather than processes, not all of them show up in the output of the **ps** command.

The **-admingroup** option is used to associate system administrators with the **fxd** process. Members of the group specified with the -admingroup option have the necessary ACL and UNIX permissions to change the permissions of any file or directory object exported from the machine. They have the equivalent of the ACL **c** permission on the objects in each exported DCE LFS fileset, and they can effectively change the mode bits on the objects in each exported non-LFS fileset. (To change the permissions on an object that resides in a lower-level directory of an exported fileset, a member of the group may need to provide the group with the necessary permissions on directories in the path that leads to the object.) Members of the group can also change the owner and owning group of any object exported from the machine. Note that, while similar in many respects, inclusion in the group specified with the **-admingroup** option and being logged in as **root** are *not* equivalent.

Place only highly trusted users in the group associated with the **fxd** process. Members of the group generally constitute a subset of the users in other DFS administrative lists such as the **admin.bos** file. For simplified administration, the same group can be specified with the **-admingroup** options of all **fxd** commands issued in a domain.

The **fxd** command includes a number of options that affect the File Exporter's management of tokens. The following two sections describe only those token-related issues germane to the **fxd** command's options. Tokens, their management by the File Exporter, and their benefits and implications are described in Part 1 of this manual.

Token Management

Token management refers to the File Exporter's use of tokens to synchronize access to data and metadata on a File Server machine. The File Exporter uses tokens to track the clients that have accessed data from the machine and the types of operations they are permitted to perform on the data. When a client wants to access or change data on a File Server machine, it contacts the File Exporter on the machine to request the necessary tokens for the data. If the File Exporter can grant the client the requested tokens, the client in turn can use the tokens to access the data from the File Exporter.

Many factors affect the File Exporter's ability to grant a client's request for tokens. The File Exporter can always grant requests for tokens that do not conflict with those already held by another client. If requested tokens do conflict with existing tokens held by another client, the File Exporter tries to revoke the existing tokens. If it can revoke the existing tokens, it grants those requested; if it cannot, it either places the request in a queue or refuses it. (The choice is the client's.)

When its tokens are revoked, a client such as the Cache Manager flushes cached data for which the tokens applied, writing any modified data back to the File Server machine. Among the factors that affect the File Exporter's ability to revoke existing tokens are the various lifetimes it associates with the tokens it grants and the clients to which it grants them. The following list briefly introduces these values, of which the latter two can be modified with options of the **fxd** command:

Token lifetime
: Specifies the length of time for which a token is valid. The File Exporter needs to revoke only valid tokens. Once a token has expired, the File Exporter does not need to revoke it; it can simply grant new tokens as if the expired token did not exist.

Host lifetime
: Specifies the length of time for which the File Exporter considers a client to be alive. A client must contact the File Exporter to renew its host lifetime before it expires. As long as a client's host lifetime has not expired, the File Exporter cannot revoke its tokens without its permission.

Host RPC lifetime
: Specifies the length of time for which the File Exporter agrees to attempt to make an RPC to a client before it revokes its tokens. The client's response to the RPC renews its host lifetime, meaning the File Exporter cannot revoke its tokens without its permission. If the client fails to respond to the RPC but its host lifetime has not expired, the File Exporter cannot revoke its tokens; if it fails to respond and its host lifetime has expired, the File Exporter can revoke any tokens it holds without contacting it further. The File Exporter can revoke a client's tokens without contacting it once its host RPC lifetime has expired. A client's host RPC lifetime must be at least as long as its host lifetime.

In general, the following rules apply to the File Exporter's revocation of valid tokens:

1. If the client's host lifetime has not expired, the File Exporter tries to contact the client; it must have the client's permission to revoke its tokens.
2. If the client's host lifetime has expired but its host RPC lifetime has not, the File Exporter tries to contact the client one time. If the client responds, the File Exporter cannot revoke its tokens without its permission; otherwise, the File Exporter can revoke any tokens held by the client without contacting it further.

3. If the client's host RPC lifetime has expired, the File Exporter can revoke its tokens without contacting it.

The following options of the **fxd** command can be used to modify the lifetimes the File Exporter assigns to its clients. By default, the File Exporter use values of 2 minutes for each of these lifetimes.

-hostlife
: Specifies each client's default host lifetime. The **-hostlife** must be greater than 0 (zero) and less than or equal to both the **-maxlife** and the **-hostrpc**.

-maxlife
: Specifies the maximum host lifetime the File Exporter will grant to a client that asks for one larger than the default specified with the **-hostlife** option. The **-maxlife** must be greater than or equal to the **-hostlife** and less than or equal to the **-maxrpc** .

-hostrpc
: Specifies each client's default host RPC lifetime. The **-hostrpc** must be greater than or equal to the **-hostlife** and less then or equal to the **-maxrpc**.

-maxrpc
: Specifies the maximum host RPC lifetime the File Exporter will grant to a client that asks for one larger than the default specified with the **-hostrpc** option. The **-maxrpc** must be greater than or equal to both the **-maxlife** and the **-hostrpc**.

If you use one of these options to modify a default lifetime value, be careful not to violate any of the dependency rules described in the previous list. In some cases, the command can adjust values not modified by the user to ensure that the dependencies are not violated, as follows:

- If you increase the value of **-hostlife** without specifying **-maxlife**, -hostrpc, or **-maxrpc**, the command increases the other three values as necessary.
- If you increase the value of **-maxlife** without specifying **-maxrpc**, the command increases the value of **-maxrpc** as necessary.
- If you increase the value of **-hostrpc** without specifying **-maxrpc**, the command increases the value of **-maxrpc** as necessary.
- If you decrease the value of **-maxlife** without specifying **-hostlife**, the command decreases the value of **-hostlife** as necessary.
- If you decrease the value of **-maxrpc** without specifying **-hostrpc**, the command decreases the value of **-hostrpc** as necessary.
- If you specify multiple values that explicitly violate one or more of the dependency rules, the command fails.
- If you specify a value that implicitly violates one or more of the dependency rules and the command cannot adjust other values to compensate for the violation, the command fails.

The command displays an appropriate message if it adjusts a value that was not specified or if it fails because specified values violate the previously defined rules.

Token State Recovery

Token state recovery refers to clients regaining their tokens following a network failure or File Server machine restart. In either of these situations, each client that cannot contact the File Exporter polls the File Exporter at regular intervals. When it

can again reach the File Exporter, the client attempts to recover tokens it had before it lost contact. The frequency with which each client tries to contact the File Exporter in these cases is defined with the **-pollinterval** option of the **fxd** command; by default, each client polls the File Exporter every 3 minutes.

In the case of a network failure, a client may be unable to prevent its host lifetime from expiring before it can again contact the File Exporter. Once communication is restored, the client must either reclaim its tokens or request new ones, as necessary. The client may need to compete for its tokens with other clients to which the tokens were granted while it could not reach the File Exporter.

In the case of a File Exporter restart, the File Exporter loses all knowledge of tokens it granted. For a brief period after it restarts, it refuses all requests for new tokens from all clients. During this period, it accepts requests only to reestablish tokens from those clients that held them before it was restarted. The File Exporter gives those clients that held tokens before it was restarted the chance to recover their tokens without having to compete with other clients that could request the same tokens.

The File Exporter bases the length of its period of token state recovery after a restart on the **-maxlife** or the **-pollinterval**, whichever is greater (it adds 20 seconds to the value it chooses to compensate for its own initialization time). The larger of these two values ensures that each client that had tokens has an opportunity to contact the File Exporter before the File Exporter accepts requests for new tokens from all clients. (Within this time, each client will contact the File Exporter either to renew its host lifetime or to poll the File Exporter.)

If the File Exporter receives many requests to reestablish tokens just prior to the end of its token state recovery period, it dynamically extends the original length of the period. If many clients continue to contact it during the extension, the File Exporter continues to extend the period incrementally, to a maximum of twice its original length.

(Note that, if a client is restarted for any reason, it loses all knowledge of the tokens it possessed prior to the restart; recovery of its tokens is not possible.)

Security

The **-minlocalprotectlevel**, **-maxlocalprotectlevel**, **-minremoteprotectlevel**, and **-maxremoteprotectlevel** options set the minimum and maximum RPC authentication bounds for communications between the File Exporter and clients. These bounds are used in negotiating an RPC authentication level for communications with clients. Two sets of bounds are maintained: a set that governs communications with clients within the same cell, and a second set that governs communications with clients in foreign cells.

In operation, the File Exporter and client (Cache Manager) interact to arrive at a mutually acceptable authentication level for communications. The negotiation starts with an RPC using the initial authentication level sent from the Cache Manager to the File Exporter. If the initial authentication level is outside the minimum or maximum bounds set through the **fxd** command, the File Exporter returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Exporter requests a security level below the minimum set at the Cache Manager (causing

the Cache Manager to refuse communications with the File Exporter). Once the Cache Manager and File Exporter have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Exporter.

In addition, administrators can also set advisory bounds on a per-fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level . Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset.

Note that the use of this command does not preclude communications with File Servers running earlier versions of DFS.

The various authentication levels are set by specifying either an integer value between 0 and 6, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **rpc_protect_level_default** or **default** or **0**: Use the DCE default authentication level.
- **rpc_protect_level_none** or **none** or **1**: Perform no authentication.
- **rpc_protect_level_connect** or **connect** or **2**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_protect_level_call** or **call** or **3**: Authenticate only at the beginning of each RPC received.
- **rpc_protect_level_pkt** or **pkt** or **4**: Ensure that all data received is from the expected host.
- **rpc_protect_level_pkt_integrity** or **pkt_integrity** or **5**: Authenticate and verify that none of the data transferred has been modified.
- **rpc_protect_level_pkt_privacy** or **pkt_privacy** or **6**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

The default values of the File Exporter and Cache Manager are such that, if they are not changed, the File Exporter and Cache Manager will negotiate to the packet level for local cell and packet integrity level for non-local cell access.. The default File Exporter values are as follows:

- The default minimum authentication level for communications with clients in the local cell is set to none.
- The default maximum authenticationn level for communications with clients in the local cell is set to packet privacy.
- The default minimum authentication level for communications with clients in foreign cells is set to none.
- The default maximum authentication level for communications with clients in foreign cells is set to packet privacy.

The default Cache Manager settings are as follows:

- The default initial authentication level for communications with File Exporters in the local cell is set to packet.

- The default minimum authentication level for communications with File Exporters in the local cell is set to none.
- The default initial authentication level for communications with File Exporters in foreign cells is set to packet integrity.
- The default minimum authentication level for communications with File Exporters in foreign cells is set to packet.

Given that both Cache Manager default initial authentication levels are set to packet and that this level is within the default bounds set at the File Exporter, the default authentication level is therefore packet. If you set the minimum bound at the File Exporter higher than packet integrity, any Cache Managers from a version of DFS previous to OSF DCE 1.2.2 (DCE 2.2 for AIX) will not be able to communicate with that File Exporter.

Privilege Required

The issuer must be logged in as **root** on the local machine.

CAUTIONS

If you restart the File Exporter with the **fxd** command's **-notsr** option, the File Exporter does not enter token state recovery; clients do not have a protected opportunity to reestablish their tokens after the restart. Similarly, if you restart the File Exporter using different values for the command's lifetime or polling interval values, the File Exporter may not remain in token state recovery long enough to provide all clients an opportunity to reestablish their tokens after it is restarted. (Until they reestablish contact with the File Exporter, clients continue to use the previous lifetime and polling interval values, which may be too long if the File Exporter is directed to use shorter values when it is restarted.)

If you set the minimum RPC authentication level for communications with clients in either local or foreign cells to higher than packet integrity, the affected clients that are running a version of DFS previous to OSF DCE 1.2.2 (DCE 2.2 for AIX) will not be able to communicate with the File Exporter.

OUTPUT

The command sends error messages to standard error output (**stderr**) if problems are encountered during initialization. It also displays error messages if you specify values for its lifetime-related options that violate the dependencies mentioned in the section on Token Management. Finally, it displays warning messages if it adjusts one or more of its lifetime values to compensate for an option you specify.

EXAMPLES

The following line, entered in the appropriate initialization file (**/etc/rc** or its equivalent) on a File Server machine, starts the **fxd** process on the local machine. The **cell_fileset** group is specified as the administrative group for the File Exporter on the machine. The **dfsbind** process must be run before the **fxd** process in a start-up file.

```
fxd -admin cell_fileset
```

The previous command line can be modified as follows to increase the host RPC lifetime, maximum host lifetime, and maximum host RPC lifetime associated with the File Exporter:

```
fxd -admin cell_fileset -hostrpc 180 -maxlife 240
```

These options change the File Exporter's lifetime values, as follows:

- The **-hostrpc** option explicitly increases the host RPC lifetime to 3 minutes.
- The **-maxlife** option explicitly increases the maximum host lifetime to 4 minutes. It also causes the command to implicitly increase the maximum host RPC lifetime to 4 minutes. (Note that, had the **-maxlife** option been omitted, the command would have implicitly increased the maximum host RPC lifetime to 3 minutes to match the increase to the host RPC lifetime.)

RELATED INFORMATION

Commands: **dfsbind(8dfs)**, **ftserver(8dfs) fts_setprotectlevels(8dfs)**.

growaggr

Purpose

Increases the size of a DCE LFS aggregate

Synopsis

growaggr -aggregate *name*[**-aggrsize** *blocks*][**-noaction**][**-help**]

OPTIONS

-aggregate *name*

Specifies the device name or aggregate name of the DCE LFS aggregate whose size is to be increased. These names are specified in the first and second fields of the entry for the aggregate in the *dcelocal*/**var/dfs/dfstab** file. The specified aggregate does not need to be exported, nor does any fileset on the aggregate need to be mounted locally or in the global namespace.

-aggrsize *blocks*

Specifies the total number of 1024-byte blocks to be available on the specified aggregate. The number of 1024-byte blocks specified with this option cannot exceed the total size of the disk partition on which the aggregate resides, and it must be at least three DCE LFS blocks greater than the current size of the aggregate. (The number of bytes in a DCE LFS block is defined on a per-aggregate basis with the **-blocksize** option of the **newaggr** command when an aggregate is created.)

Include the **-noaction** option with this option to determine if the specified aggregate size is valid without changing the current size of the aggregate. Omit both this option and the -noaction option to increase the size of the aggregate to the total size of the disk partition on which it resides.

-noaction

Used without the **-aggrsize** option, this option directs the command to display the total number of 1024-byte blocks on the disk partition on which the specified aggregate resides. Used with the **-aggrsize** option, this option determines if the specified aggregate size is valid. The current size of the specified aggregate is not affected if this option is used.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **growaggr** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The **growaggr** command is used to increase the size of an existing DCE LFS aggregate. The aggregate whose size is to be increased is specified with the **-aggregate** option. The binary file for the **growaggr** command resides in *dcelocal*/**bin/growaggr**.

The **-aggrsize** option is used to specify the total size to make the aggregate. Specify the size as a number of 1024-byte blocks. The size specified with this option cannot exceed the total size of the disk partition on which the aggregate

resides. The specified size also must be at least three DCE LFS disk blocks greater than the current size of the aggregate. If it is not, the command displays the minimum size in 1024-byte blocks that can be specified. (The number of bytes in a DCE LFS block is defined on a per-aggregate basis with the **-blocksize** option of the **newaggr** command when an aggregate is initialized. It must be a power of 2 between 1024 and 65,536.)

If the **-noaction** option is included with the command, the present size of the aggregate is not affected. Combine the **-aggrsize** and **-noaction** options to achieve the following results:

- Specify only the **-aggrsize** option to increase the size of the aggregate to the specified size, as described previously.
- Specify only the **-noaction** option to determine the total number of 1024-byte blocks on the partition on which the aggregate resides.
- Specify both the **-aggrsize** and **-noaction** options to determine if the size specified with the **-aggrsize** option is valid (within the limits defined previously).
- Omit both the **-aggrsize** and **-noaction** options to increase the size of the aggregate to the total size of the disk partition on which it resides.

In operating systems that support logical volumes, the command is useful for increasing the size of an aggregate when the size of the logical volume on which the aggregate resides is increased. It can also be used to increase the size of an aggregate that was deliberately made smaller than the size of the partition or logical volume on which it resides.

On DCE 2.2 for AIX, first increase the size of the AIX logical volume and then run **growaggr** to increase the size of the DCE LFS aggregate.

The command does not affect any data or filesets that already reside on the aggregate to be grown.

Privilege Required

If the **-noaction** option is *not* included with the command, the issuer must have both the read and write permissions for the device (disk partition) on which the specified aggregate resides; if the **-noaction** option is included with the command, the issuer needs only the read permission for the device on which the aggregate resides. An issuer who is logged in as **root** on the machine on which the aggregate resides always has the necessary privilege to issue this command.

RELATED INFORMATION

Commands: **newaggr(8dfs)**.

Files: **dfstab(4dfs)**.

newaggr

Purpose

Initializes a DCE LFS aggregate

Synopsis

newaggr -aggregate *name* **-blocksize** *bytes* **-fragsize** *bytes* [**-initialempty** *blocks*] [**-aggrsize** *blocks*] [**-logsize** *blocks*] [**-overwrite] [-verbose] [-noaction] [-help]**

OPTIONS

-aggregate *name*
Specifies the device name or aggregate name of the disk partition to be initialized as a DCE LFS aggregate. These names are specified in the first and second fields of the entry for the aggregate in the *dcelocal*/**var/dfs/dfstab** file.

-blocksize *bytes*
Specifies the number of bytes to be available in DCE LFS blocks on the aggregate (also referred to as the blocking factor). The value provided must be a power of 2 between 1024 and 65,536.

The number controls how disks are addressed in DCE LFS. No file larger than 2^{31} blocks can be read or written. (Other considerations, chiefly I/O speed versus disk utilization, also constrain the maximum file size.)

-fragsize *bytes*
Specifies the number of bytes to be available in DCE LFS fragments on the aggregate. The value provided must be a power of 2 between 1024 and the number of bytes specified with **-blocksize**.

The unit of storage allocation in DCE LFS is the fragment, so this value controls the granularity of storage allocated to files. In other words, it affects the amount of space lost due to fragmentation.

-initialempty *blocks*
Specifies the number of DCE LFS blocks that DCE LFS leaves empty at the beginning of the disk partition when it initializes the aggregate. The value provided must be an integer between 0 (zero) and 65,536 divided by the number of bytes specified with **-blocksize**. For example, if the value provided with **-blocksize** is 2048, the value provided with **-initialempty** cannot exceed 32 (65,536 divided by 2048).

The empty blocks reserved with this option are often used for a bootstrapping program. For this reason, the reserved blocks are often referred to as *bootblocks*.

If this option is omitted, one block is left empty at the beginning of the partition.

-aggrsize *blocks*
Specifies the total number of DCE LFS blocks to be available on the aggregate. Because this value cannot exceed the size of the disk partition, it can be used only to restrict the size of the aggregate. It must be large enough to accommodate at least the log and any blocks left empty at the beginning of the partition.

If this option is omitted, the default is the total number of DCE LFS blocks on the disk partition being initialized as a DCE LFS aggregate.

-logsize *blocks*
Specifies the number of DCE LFS blocks to be reserved for the log on the aggregate. This value cannot exceed the number of DCE LFS blocks used for **-aggrsize**, and it must contain at least enough blocks for the log to be initially created.

If this option is omitted, the default is to reserve 50 DCE LFS blocks for the log. However, 50 blocks is only the default; the command never reserves less than 1% or more than 10% of the total number of DCE LFS blocks on the aggregate (the number of blocks specified with the **-aggrsize** option) for the log. The maximum log size is 16 MB.

-overwrite
Specifies that any existing file system found on the partition can be overwritten when the aggregate is initialized. If this option is specified, an existing file system on the disk partition is automatically overwritten; the issuer is not prompted for confirmation.

If this option is omitted and an existing file system is found on the partition, the command displays a message informing the issuer that the **-overwrite** option must be used to overwrite an existing file system. It then terminates with an exit code of at least 16 without overwriting the existing file system.

-verbose
Directs the command to provide more information on its actions as it executes. The information is displayed on standard output (**stdout**) unless it is directed elsewhere.

-noaction
Directs the command to display information about what it would do without actually modifying the partition. Include the other options as you would to actually execute the command. The command displays the default values it would use for its options and informs the issuer if the disk partition already contains a file system.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **newaggr** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The **newaggr** command is used to initialize a partition on the local disk of a machine for use as an aggregate with DCE LFS. The partition to be initialized as a DCE LFS aggregate is specified with the **-aggregate** option. The **newaggr** command formats the specified partition by creating the metadata structure used by DCE LFS for access control list (ACL) support, logging, and multiple fileset operations. It also creates temporary space on the disk used by the DCE LFS log for faster restarts after system failures. The binary file for the **newaggr** command resides in *dcelocal***/bin/newaggr**.

An aggregate is a collection of DCE LFS disk blocks made up of the space available on the partition on which it resides. Each disk block on an aggregate has a fixed size specified with the **-blocksize** option. The **-blocksize** option specifies

the number of bytes in each DCE LFS block. The value specified with this option must be a power of 2 between 1024 (1 kilobyte) and 65,536 (64 kilobytes).

Each block can be further decomposed into fragments. Each fragment on an aggregate has a fixed size specified with the **-fragsize** option. The **-fragsize** option specifies the number of bytes in each fragment. The value specified with this option must be a power of 2 between 1024 (1 kilobyte) and the value specified with the **-blocksize** option.

DCE LFS manages blocks and fragments as variable-length containers for the storage of user and system data. It manages filesets created on the aggregate as logically independent collections of data. Each fileset consists of a hierarchical collection of files residing entirely within a single aggregate. DCE LFS obtains blocks for each fileset from a common allocation pool. As a result, filesets can share blocks (if the blocks are copy-on-write or if each fileset uses only a fragment of the block).

The **-initialempty** option can be used to reserve a number of empty blocks at the beginning of a partition. The empty blocks are referred to as *bootblocks* because they are often used for bootstrapping programs. The value provided with the **-initialempty** option must be an integer between 0 (zero) and 65,536 divided by the value specified with the **-blocksize** option. By default, one block is left empty.

The **-aggrsize** option can be used to restrict the number of DCE LFS blocks in the aggregate. By default, all of the blocks available on the disk partition to be initialized are used in the aggregate. The value specified with the **-aggrsize** option cannot exceed the size of the partition being initialized. It must be large enough to accommodate at least the log and any blocks left empty at the beginning of the partition.

The **-logsize** option can be used to specify the number of DCE LFS blocks to be reserved for the log on the aggregate. By default, the command reserves 50 DCE LFS blocks for the log, but it never reserves less than 1% or more than 10% of the total size of the aggregate for the log. The value specified with the **-logsize** option cannot exceed the number of DCE LFS blocks used for the **-aggrsize** option, and it must specify at least enough blocks for the log to be initially created. The maximum log size is 16 MB.

DCE LFS also reserves a variable amount of disk space on the aggregate. By default, DCE LFS reserves 2 megabytes of disk space on an aggregate. However, no less than 1% or no more than 10% of the total size of an aggregate is ever reserved; for example, only 1.5 megabytes are reserved on an aggregate whose total size is only 15 megabytes.

Reserved disk space is used for internal purposes. For example, the reserved space is used to avoid potential problems with routine administrative operations such as fileset moves and clones. The reserved space is not directly accessible to users and administrators. Use the **fts aggrinfo** command to display the total amount of disk space, including the amount of reserved disk space, on an aggregate.

If an existing file system on the disk partition being initialized is to be overwritten, include the **-overwrite** option with the command. The option instructs the command to overwrite any data found on the partition. To prevent an existing file system from being overwritten, omit the **-overwrite** option. If the command encounters an existing file system, it stops the initialization procedure without overwriting the

existing file system and reports that it found a file system on the partition. It also instructs you to include the **-overwrite** option with the command to overwrite the resident file system.

Use the **-noaction** option to have the command report whether the partition already contains a file system or to display the values it calculates for the **-aggrsize** and **-logsize** options without actually overwriting a file system or initializing the partition. Specify all of the command's options as you would to actually execute the command, and include the **-noaction** option to display the results of the command without modifying the partition.

The **newaggr** command must be used to initialize a disk partition before the partition can contain DCE LFS filesets. After the disk partition is initialized as a DCE LFS aggregate with this command, an entry can be created for the aggregate in the **dfstab** file, and it can be exported to the DCE namespace with the **dfsexport** command. DCE LFS filesets can then be created on it with the **fts create** command and mounted in the global namespace with the **fts crmount** command.

Because the **newaggr** command overwrites all data on the partition being initialized, the partition must not be mounted locally and it should not contain data when the command is run. If the **newaggr** command is issued with the **-overwrite** option to create a DCE LFS aggregate on a disk partition that already contains a file system, the previous file system is destroyed. However, the command fails if it is run on an aggregate or partition that is currently exported to the DCE namespace, or if it is run on an aggregate that houses a locally mounted fileset. (If necessary, the **dfsexport** command can be used to detach an aggregate or partition from the namespace.)

In operating systems that support logical volumes, the command can be used to initialize a logical volume as a DCE LFS aggregate. In such cases, all of the command's functionality described here with respect to a disk partition applies to the logical volume.

CAUTIONS

Do not use the **newaggr** command to create nonLFS aggregates. Do not use the command on a partition that contains data you want to retain; the command destroys all data on any partition it initializes. Do not use the command on a locally mounted partition; doing so causes the kernel to panic. Finally, do not use the command on a currently exported aggregate or partition, or on an aggregate that houses a locally mounted fileset; the command fails in these cases.

Privilege Required

If the **-noaction** option is *not* included with the command, the issuer must have both the read and write permissions for the device (disk partition) to be initialized as a DCE LFS aggregate; if the **-noaction** option is included with the command, the issuer needs only the read permission for the specified device. An issuer who is logged in as **root** on the machine on which the specified device resides always has the necessary privilege to issue this command.

RELATED INFORMATION

Commands: **dfsexport(8dfs)**, **fts aggrinfo(8dfs)**, **growaggr(8dfs)**.

Files: **dfstab(4dfs)**.

repserver

Purpose

Initializes the Replication Server process

Synopsis

repserver [**-mainprocs** *number_of_background_daemons*][**-tokenprocs** *number_of_token_daemons*] [**-help**]

OPTIONS

-mainprocs *number_of_background_daemons*
Specifies the number of background daemons to run on the machine. These daemons are responsible for the bulk of the effort required to maintain read-only replicas on the local machine, as well as for receiving and servicing RPC requests from DFS clients. If this option is omitted, four background daemons perform these services.

-tokenprocs *number_of_token_daemons*
Specifies the number of background daemons dedicated to servicing token revocation RPC requests from File Exporters. If this option is omitted, four background daemons service token revocation requests.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **repserver** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The Replication Server, or **repserver** process, in conjunction with the Cache Manager, tracks the currency of replicas and updates the versions of data being used at each replication site. The **repserver** process is used in Release and Scheduled Replication, and must run on any machine that stores read-only replicas of read/write filesets. For simplified administration, run the **repserver** process on all File Server machines. The **repserver** process is usually started and controlled by the BOS Server; if it is not, execute the **repserver** process as a background process. The binary file for the **repserver** process resides in *dcelocal***/bin/repserver**.

The **-mainprocs** and **-tokenprocs** options can be used to alter the default number of background daemons running on the server machine, as follows:

-mainprocs
Specifies the number of background daemons that run on the machine to maintain read-only replicas housed on the local machine and to service RPC requests from DFS clients. The default number of background daemons is four. Use the **-mainprocs** option to increase the number of background daemons if the machine houses a large number of replicas.

-tokenprocs
Specifies the number of background daemons dedicated to handling token revocation RPC requests from the File Exporters on File Server machines. The default number of background daemons dedicated to this task is four. If

the **-mainprocs** option is used to increase the number of background daemons dedicated to maintaining replicas and servicing RPC requests from DFS clients, use the **-tokenprocs** option to increase the number of background daemons dedicated to servicing token revocation requests from File Exporters.

When it is started, **repserver** creates the *dcelocal***/var/dfs/adm/RepLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when **repserver** is started, the process moves it to the **RepLog.old** file in the same directory (overwriting the current **RepLog.old** file if it exists) before creating a new version to which to append messages.

Use the **fts statrepserver** command to check the status of the Replication Server on any server machine. Use the **fts lsreplicas** command to check the status of fileset replicas.

Privilege Required

The issuer must be logged in as **root** on the local machine.

OUTPUT

If problems are encountered during initialization, **repserver** sends error messages to standard error output (**stderr**). The **repserver** process keeps an event log in *dcelocal***/var/dfs/adm/RepLog**.

RELATED INFORMATION

Commands: **fts lsreplicas(8dfs)**, **fts statrepserver(8dfs)**.

Files: **RepLog(4dfs)**.

salvage

Purpose

Uses the DFS Salvager to recover, verify, or salvage the structure of a DCE LFS aggregate

Synopsis

salvage -aggregate *name*[**-recoveronly][{-verifyonly | -salvageonly }][-force] [-verbose][-help]**

OPTIONS

-aggregate *name*
: Specifies the device name or aggregate name of the DCE LFS aggregate to be verified, recovered, or salvaged. These names are specified in the first and second fields of the entry for the aggregate in the *dcelocal***/var/dfs/dfstab** file.

-recoveronly
: Directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate. See the Description section for information about using and combining the command's options.

-verifyonly
: Directs the Salvager to verify the specified aggregate. The Salvager examines the structure of the aggregate to determine if it contains any inconsistencies, reporting any that it finds. See the Description section for information about using and combining the command's options.

-salvageonly
: Directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate. See the Description section for information about using and combining the command's options.

-force Executes the Salvager in noninteractive mode. By default, the Salvager prompts for confirmation before proceeding in certain situations (for example, if it believes an aggregate on which it is run may be a nonLFS partition). Use this option to direct the Salvager to proceed with all operations without asking whether it should continue. Use this option with care; the Salvager's changes can be unpredictable if this option is used with an invalid aggregate.

-verbose
: Directs the Salvager to produce detailed information about the aggregate as it executes. The information is useful primarily for debugging purposes. It is displayed on standard output (which can be redirected). Use this option alone or with any combination of the available options.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **salvage** command. See the **bos help** and **bos apropos** reference pages for examples using these commands.

DESCRIPTION

The *dcelocal*/**bin/salvage** command invokes the DFS Salvager on the DCE LFS aggregate specified with the **-aggregate** option. Following a system restart, the Salvager employs the DCE LFS log mechanism to return consistency to a file system by running recovery on the aggregate on which the file system resides. Recovery is the replaying of the log on the aggregate; the log records all changes made to metadata as a result of operations such as file creation and deletion. If problems are detected in the basic structure of the aggregate, if the log mechanism is damaged, or if the storage medium of the aggregate is suspect, the **salvage** command must be used to verify or repair the structure of the aggregate.

Use the command's **-recoveronly**, **-verifyonly**, and **-salvageonly** options to indicate the operations the Salvager is to perform on the specified aggregate, as follows:

- Specify the **-recoveronly** option to run recovery on the aggregate without attempting to find or repair any inconsistencies found on it. Recovery is the replaying of the log on the aggregate. Use this option to quickly return consistency to an aggregate that does not need to be salvaged; this represents the normal production use of the Salvager. Unless the contents of the log or the physical structure of the aggregate is damaged, replaying the log is an effective guarantee of a file system's integrity.
- Specify the **-verifyonly** option to determine whether the structure of the aggregate contains any inconsistencies without running recovery or attempting to repair any inconsistencies found on the aggregate. Use this option to assess the extent of the damage to an aggregate. The Salvager makes no modifications to an aggregate during verification. Note that it is normal for the Salvager to find errors when it verifies an aggregate that has not been recovered; the presence of an unrecovered log on an aggregate makes the findings of the Salvager, positive or negative, of dubious worth.
- Specify the **-recoveronly** and **-verifyonly** options to run recovery on the aggregate and then analyze its structure without attempting to repair any inconsistencies found on it. Use these options if you believe replaying the log can return consistency to the aggregate, but you want to verify the consistency of the aggregate after recovery is run. Recovering an aggregate and then verifying its structure represents a cautious application of the Salvager.
- Specify the **-salvageonly** option to attempt to repair any inconsistencies found in the structure of the aggregate without first running recovery on it. Use this option if you believe the log is damaged or replaying the log will not return consistency to the aggregate and may in fact further damage it. Under normal circumstances, do not salvage an aggregate without first recovering it.
- Omit the **-recoveronly**, **-verifyonly**, and **-salvageonly** options to run recovery on the aggregate and then attempt to repair any inconsistencies found in the structure of the aggregate. Because recovery eliminates inconsistencies in an undamaged file system, an aggregate is typically recovered before it is salvaged. In general, it is good first to recover and then to salvage an aggregate if a machine panics or experiences a hardware failure.

  Omit these three options if you believe the log should be replayed before attempts are made to repair any inconsistencies found on the aggregate. (Omitting the three options is equivalent to specifying the **-recoveronly** and **-salvageonly** options.)

The following rule summarizes the interaction of the **-recoveronly**, **-verifyonly**, and **-salvageonly** options: The **salvage** command runs recovery on an aggregate and

attempts to repair it *unless* one of the three options is specified; once one of these options is specified, you must explicitly request any operation you want the Salvager to perform on the aggregate.

The basic function of the Salvager is similar to that of the UNIX **fsck** program. The Salvager recovers a DCE LFS aggregate and repairs problems it detects in the structure of the aggregate. It does not verify or repair the format of user data contained in files on the aggregate. If it makes changes, the Salvager displays the pathnames of the files affected by the modifications, when the pathnames can be determined. The owners of the files can then verify the files' contents, and the files can be restored from backups if necessary.

The Salvager verifies the structure of an aggregate by examining all of the anodes, directories, and other metadata in each fileset on the aggregate. An anode is an area on the disk that provides information used to locate data such as files, directories, ACLs, and other types of file system objects. Each fileset contains an arbitrary number of anodes, all of which must reside on the same aggregate. By following the links between the various types of anodes, the Salvager can determine whether the organization of an aggregate and the filesets it contains is correct and make repairs if necessary.

Not all aggregates can be salvaged. In cases of extensive damage to the structure of the metadata on an aggregate or damage to the physical disk that houses an aggregate, the Salvager cannot repair inconsistencies. Also, the Salvager cannot verify or repair damage to user data on an aggregate. The Salvager cannot detect problems that modified the contents of a file but did not damage the structure of an aggregate or change the metadata of the aggregate.

Like the UNIX **fsck** command, the Salvager analyzes the consistency of an aggregate by making successive passes through the aggregate. With each successive pass, the Salvager examines and extracts a different type of information from the blocks and anodes on the aggregate. Later passes of the Salvager use information found in earlier passes to help in the analysis.

Unlike the **fsck** command, the Salvager does not normally prompt for additional information as it executes. It typically performs the requested operation without prompting for input or pausing to verify any changes before it makes them. It prompts for confirmation only in the following cases:

- It believes the specified aggregate does not contain a DCE LFS file system. This can occur if it finds a nonLFS superblock whose creation time is more recent than the creation time of the DCE LFS superblock.
- It finds that the size of the aggregate recorded in the DCE LFS superblock exceeds the capacity of the partition on which the aggregate resides.

At the prompt, you can choose to cancel or continue the operation. If you continue the operation under either of these circumstances and the aggregate proves to be invalid, unpredictable results can ensue. The best response in either case is to cancel the operation and attempt to determine the cause of the problem.

If you are confident that you want the Salvager to continue in any case, you can include the **-force** option with the command. This option directs the Salvager to perform the requested operation without prompting for confirmation. Exercise caution when using the **-force** option; the Salvager can produce unpredictable results if this option is used with an invalid aggregate.

In general, the Salvager exits with an error code of at least **16** without analyzing a partition that it is sure is not a DCE LFS aggregate. It also exits with an error code of **16** if an aggregate to be recovered or salvaged is currently exported to the global namespace, or if a fileset on the aggregate to be recovered or salvaged is mounted locally. (If necessary, you can use the **dfsexport** command to detach an exported aggregate from the namespace.)

As the Salvager executes, it maintains a number of internal lists. Each list consists of anodes that failed verification in specific ways. When it initially scans an aggregate, the Salvager marks as ″unsafe″ anodes with which it encounters problems. The Salvager later attempts to determine the actual pathnames associated with these anodes to include the pathnames in the lists. When it has finished salvaging, the Salvager displays any nonempty lists. It also returns one of a number of informative exit codes, depending on the inconsistencies it found and the repairs it made. More information about the lists and exit codes displayed by the Salvager appears later in this reference page.

Internal structures maintained by the Salvager require a minimum of 1 megabyte of swap space. However, the total amount of swap space required by the Salvager depends largely on the size of the aggregate being salvaged and the extent of the damage to the aggregate.

Privilege Required

The privileges required depend on whether the **-recoveronly**, **-verifyonly**, or **salvageonly** option is specified with the command: If just the -verifyonly option is included, the issuer needs only the read permission for the specified device (aggregate); if the **-recoveronly** or **-salvageonly** option is included, or if all three of these options are omitted, the issuer must have both the read and write permissions for the specified device. An issuer who is logged in as **root** on the machine on which the specified device resides always has the necessary privilege to issue the command.

CAUTIONS

The Salvager can be used to salvage only DCE LFS aggregates. If it is executed on a nonLFS partition, it exits with an error code of at least **16** without performing any operations. Use the UNIX **fsck** program or its equivalent to verify or restore consistency to nonLFS disk partitions.

By default, the Salvager asks for confirmation before proceeding with operations on aggregates that it suspects are nonLFS partitions or whose indicated sizes exceed the capacities of the partitions on which they reside. The command's -force option can be used to direct the Salvager to continue without prompting in these cases. Do not include the **-force** option under normal conditions; the Salvager can make undesirable changes if the option is used with an invalid aggregate.

If the Salvager is used to recover or salvage an aggregate that is currently exported, it exits with an error code of **16** without performing the operation. Use the **dfsexport** command to detach an aggregate from the global namespace if necessary before recovering or salvaging it. (The Salvager can be used to verify the structure of a currently exported aggregate, but this is not a good practice; the results may be misleading.) The Salvager also exits with an error code of **16** if a fileset on an aggregate to be recovered or salvaged is mounted locally.

OUTPUT

The Salvager sends output to both **stdout** and **stderr**. When it is started, the Salvager displays the device name of the aggregate on which it is run and the operation it is to perform. For example, the Salvager displays the following message if it is directed to recover an aggregate:

```
Will run recovery on device
```

Similarly, the Salvager displays the following message if it is directed to verify an aggregate:

```
Verifying device
```

If you specify the **-verbose** option with the command, the Salvager generates the following information about the aggregate:

- Physical information about the configuration of the aggregate
- Header information from the aggregate, including the major and minor number of the device on which the aggregate was created, and the date and time at which the aggregate was created
- Information about how space in the aggregate is allocated, including
  - The total size of the aggregate in blocks
  - The block size
  - The fragment size
  - The number of the first block in the aggregate
  - The location of the principal superblock for the aggregate
  - The number of logical blocks in the aggregate

If you use the Salvager to recover an aggregate and the log on the aggregate does not need to be replayed, the Salvager displays only the introductory message described previously. If the log does need to be replayed and the Salvager can successfully recover the aggregate, the Salvager displays the following messages:

```
Recovery statistics
   statistics
Ran recovery on device
```

In the output, *statistics* consists of a few lines of information about the log and its replaying, and *device* is the device name of the aggregate. If it cannot run recovery for any reason, the Salvager displays an appropriate exit code. (All Salvager exit codes are listed at the end of this section.)

The Salvager can display much more output if it is asked to verify or salvage an aggregate on which it finds metadata errors. As it verifies or salvages a damaged aggregate, it displays a message similar to the following for each fileset in which it encounters metadata problems:

```
In volume fileset (avl #integer)
   in anode (#integer)
      description
```

It displays the first line once for each fileset, repeating the second and third lines once for each problem anode in the fileset. The output provides the following information:

fileset The name and ID number of each affected fileset.

avl # *integer*
A pointer to the anode for the fileset.

in anode (# *integer***)**
A pointer to the anode for a file or other object in the fileset.

description
A brief description of the problem the Salvager found with the anode. If it was used to salvage the aggregate, the Salvager also describes any actions it took to repair the anode.

When it has finished executing, the Salvager lists each file whose metadata it found to be damaged, many of which it likely repaired if it salvaged the aggregate. For each file, it displays a line of the form

```
condition   fileset:pathname volume index: integer anode index: integer
```

The output provides the following information:

condition
A string that describes the state of the file or its metadata. (Information about the possible conditions follows this list.)

fileset The name of the fileset in which the affected file resides. In some cases, the Salvager cannot determine the fileset name.

pathname
The pathname of the file, relative to the root directory of the fileset. In some cases, the Salvager cannot reconstruct the pathname for a file.

volume index
A pointer to the anode for the fileset. (This information can be used to identify earlier message displayed by the Salvager that are related to this file.)

anode index
A pointer to the anode for the file. (This information can be used to identify earlier message displayed by the Salvager that are related to this file.)

The following conditions accompany the files most in need of attention:

oughtRestore
Files in which one or more block references in the associated anode were removed or changed. Because it is unlikely such files contain all of their original data, these files should be restored from existing backups. This condition applies only to files on salvaged aggregates.

mayRestore
Files to which modifications were made (for example, files whose ACLs or property lists were changed). The owners of these files should verify their contents, or a system administrator should simply restore them from backups if a directory listing indicates that they have not been modified since the last backup was made. This condition also applies only to files on salvaged aggregates.

zeroLinkCnt
Files whose link counts should be 0 (zero). These files were deleted but not closed when the system crashed or were orphaned by the Salvager as it made repairs to the file system. The system will delete them when the aggregate is exported.

badLinkCnts
Files whose link counts were inconsistent with the number of references found to them. These files should be examined, if possible, or simply restored.

The Salvager can list a file more than once if it determines that multiple conditions apply to the file. It can also display one or more additional conditions (such as **badAcls** or **badPlists**), but files with which the additional conditions are associated are typically already covered by one or more of the conditions just described. Information in the additional lists is useful primarily for debugging purposes.

The Salvager also returns one of various exit codes to summarize its actions and findings. It returns the exit codes in the form of bits, which it uses to indicate the state of the aggregate. It can set multiple bits, but, in general, the higher the bit, the greater the severity of the aggregate's problems. (The higher bit always takes precedence when interpreting the output.) The Salvager can return the following exit codes:

All bits off
The Salvager found no problems. It displays a message that includes **Done** and **Checks out**. The command need not be run again.

First bit (0x1) set
The Salvager found one or more problems. It displays a message that includes **Done** and **Some inconsistencies found**. Run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Second bit (0x2) set
The Salvager found one or more problems and fixed them. It displays a message that includes **Done** and **Some inconsistencies repaired**. The command need not be run again. (Note that if the second bit is set, the first bit is usually also set; because the higher bit takes precedence, you do not need to run the command again.)

Third bit (0x4) set
The Salvager found one or more problems and fixed some of them. It displays a message that includes **Incomplete** and **Some repairs made**. Some problems were more severe and require a subsequent salvage to be repaired; run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Fourth bit (0x8) set
The Salvager found the aggregate to be irreparably damaged. It displays a message that begins **Problem**. Use the **newaggr** command to reinitialize the aggregate, and reconstruct the data from existing backups if possible.

Fifth bit (0x10) set
The Salvager found some serious problem that prevents it from running on the aggregate; for example, the attempted recovery of the aggregate failed because of damage to the log, or the attempted salvage of the aggregate failed because the aggregate is not a DCE LFS aggregate, it is currently exported, or it contains a locally mounted fileset. The Salvager displays a message that begins **Problem**. Attempt to determine the cause of the problem.

Including the **-verbose** option with the command produces more detailed information about the aggregate as the command executes. However, the additional information is useful primarily for debugging purposes.

EXAMPLES

The following command instructs the Salvager to recover the DCE LFS aggregate whose device name is **/dev/lv01**. This example represents the most-common application of the Salvager.

```
# salvage /dev/lv01 -recover
```

The following command instructs the Salvager to analyze the structure of the aggregate to determine if it contains any inconsistencies without running recovery or attempting to repair the inconsistencies:

```
# salvage /dev/lv01 -verify
```

The following command directs the Salvager to repair any inconsistencies it finds on the aggregate without first running recovery:

```
# salvage /dev/lv01 -salvage
```

RELATED INFORMATION

Commands: **dfsexport(8dfs)**, **newaggr(8dfs)**.

Files: **dfstab(4dfs)**.

scout

Purpose

Initializes the **scout** program

Synopsis

scout -server *machine*...**-basename** *common_prefix*][**-host**][**-frequency** *seconds*]
[**-attention** *stat/threshold_pair*...**-debug** *filename*][**-help**]

OPTIONS

-server *machine*
Specifies each File Server machine whose File Exporter is to be monitored. Use one of the following to indicate each File Server machine:

- The machine's DCE pathname (for example, */.../abc.com/hosts/fs1*). If you use the **-basename** option to specify a pathname prefix common to all machines to be monitored, you need to provide only the unique suffix of each machine name; you can omit the common DCE pathname prefix.
- The machine's host name (for example, **fs1.abc.com** or **fs1**).
- The machine's IP address (for example, **11.22.33.44**).

-basename *common_prefix*
Specifies the DCE pathname prefix (for example, */.../abc.com/hosts*) common to all File Server machines specified with the **-server** option. Do not include the **/** (slash) that separates the prefix from the unique part of each machine name; it is included automatically with the **-basename** option. The basename, if specified, is displayed in the banner line.

Use this option only if you are specifying the DCE pathname of each File Server machine to be monitored. Omit this option if you are specifying the host names or IP addresses of one or more machines.

-host Displays the name of the machine running the **scout** program in the banner line. This is useful if you are logged into the machine remotely. By default, **scout** does not display this name.

-frequency *seconds*
Indicates how often the **scout** program is to probe the File Exporters. Specify a positive integer as a value in seconds; the default is 60 seconds.

-attention *stat/threshold_pair*
Specifies a list of attention settings (statistic and threshold value pairs). The **scout** program highlights any value for a statistic that exceeds its specified threshold; the highlighting is removed when the value goes below the threshold. The pairs can appear in any order. Legal statistic/threshold pairs are

conn *connections*
The maximum number of connections that principals can have open to the File Exporter before the value is highlighted. Enter a threshold for this statistic in the form of a positive integer.

fetch *number_of_fetches*
The maximum number of fetches (requests to send data) the File Exporter can service before the value is highlighted. Enter a

threshold for this statistic in the form of a positive integer. The highlighting is removed when the File Exporter is restarted, at which time the value returns to **0** (zero).

store *number_of_stores*
The maximum number of stores (requests to store data) the File Exporter can accept before the value is highlighted. Enter a threshold for this statistic in the form of a positive integer. The highlighting is removed when the File Exporter is restarted, at which time the value returns to **0** (zero).

ws *active_client_machines*
The maximum number of active client machines the File Exporter can serve before the value is highlighted. *Active* indicates those machines that communicated with the File Exporter in the past 15 minutes. Enter a threshold for this statistic in the form of a positive integer.

disk *percent_full***%**
The maximum percentage of an aggregate that can contain data before the value is highlighted. This threshold is applied to all exported aggregates and partitions on a File Server machine being monitored. Legal thresholds are the integers between 0 (zero) and 99; the default is 95%. *You must enter the % (percent sign) with this threshold.* If the % (percent sign) is absent, **scout** interprets the number as a number of kilobyte blocks. Use this threshold or use **disk** *minimum_blocks_free*.

disk *minimum_blocks_free*
The minimum number of kilobyte blocks that must be available on an aggregate before the value is highlighted. This threshold is applied to all exported aggregates and partitions on a File Server machine being monitored. Enter a threshold for this statistic in the form of a positive integer. Use this threshold or use **disk** *percent_full***%**.

-debug *filename*
Enables debugging output and directs it to the specified *filename*. Provide a complete pathname for *filename*; the current working directory is used by default. If this option is omitted, no debugging output is written.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **scout** command. See the **bos help** and **bos apropos** reference pages for examples of these commands.

DESCRIPTION

The **scout** command displays statistics gathered from the File Exporter running in the kernel on each File Server machine specified with the **-server** option. Usage statistics are also displayed about exported aggregates and partitions on the File Server machine being monitored. The **scout** program can be run on any DFS client or server machine. The binary file for the program resides in *dceshared***/bin/scout**.

To change attention settings (statistic and threshold pairs), you must stop and restart the **scout** program. In addition, **scout** does not store the settings from previous executions; you must specify the desired settings each time you start the program.

Both terminals and windowing systems that emulate terminals can display **scout** statistics. The **scout** display uses reverse video and cursor addressing; therefore, the display environment must support these features. The issuer must set the **TERM** environment variable to the correct terminal type or to one with similar characteristics.

To stop the **scout** program, enter the interrupt command (**<Ctrl-c>** or its equivalent) for your operating system in the **scout** window.

The **scout** program can display statistics in either a dedicated window or on a plain screen if a windowing environment is unavailable. The **scout** screen has three main parts: the Banner Line, the Statistics Display Region, and the Message/Probe Line.

The Banner Line at the top of the window or screen displays the word **Scout**, indicating the program is running. The name of the machine executing **scout** is displayed if the **-host** option is specified, and the basename of the File Server machines being monitored is displayed if the **-basename** option is specified.

The Statistics Display Region displays the statistics **scout** has gathered for each File Exporter. The region is divided into six columns, one column for each of the five statistic and threshold pairs used with the -attention option, and one column for the name of each File Server machine being monitored. In addition to highlighting any value that exceeds its specified attention threshold, **scout** highlights the name of any File Server machine whose File Exporter fails to respond to **scout**'s probes. The name remains highlighted until the machine resumes responding to **scout**'s probes.

The Message/Probe Line at the bottom of the window or screen indicates how many times **scout** has probed the File Exporters for statistics. Use the **-frequency** option to specify how frequently **scout** is to probe the File Exporters.

Privilege Required

No privileges are required.

EXAMPLES

The following **scout** command causes the program to monitor the File Exporters on File Server machines **fs1** and **fs2** in the cell *abc.com*. The **scout** program probes the File Exporters every 30 seconds and writes debugging information to the file named **scout.one** in the current working directory.

```
$ scout -server fs1 fs2 -basename /.../abc.com/hosts -frequency 30 -debug scout.one
```

The following command causes **scout** to monitor the same two machines. The **scout** program highlights an entry in the **Fetch** column if the File Exporter services 20,000 or more fetches, and it highlights an entry in the **Store** column if the File Exporter accepts 10,000 or more stores.

```
$ scout -server fs1 fs2 -b /.../abc.com/hosts -attention fetch 16 store 8
```

udebug

Purpose

Displays Ubik status information relevant to the specified DFS database server

Synopsis

udebug -rpcgroup *RPC_server_group* **-server** *machine*][**-long**][**-help**]

OPTIONS

-rpcgroup *RPC_server_group*
Specifies the RPC server group of the Ubik database servers whose status information you want to display. By convention, this is **/.:/fs** for the **flserver** processes and **/.:/subsys/dce/dfs/bak** for the **bakserver** processes.

-server *machine*
Names the machine containing the database server whose Ubik status information is to be displayed; if a machine name is omitted, the command uses the name of the local machine. Specify the server machine using the full DCE pathname, abbreviated host name, or IP address.

-long Directs the command to provide additional information about the other database servers in the specified RPC server group. This flag is *not* necessary if the server specified with the **-server** option is the Ubik synchronization site because the information about the other database servers is provided automatically.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

DESCRIPTION

The **udebug** command displays Ubik status information on the specified server in the specified RPC server group. If the specified server is the synchronization site or the **-long** flag is used with the command, the command displays information on all of the servers in the RPC server group.

Privilege Required

No privileges are required.

OUTPUT

The output for the **udebug** command always provides the following information for the specified database server:

- The IP address of the specified server machine. In the first example, this is **192.56.207.146**.
- The difference in seconds between the clock on the specified server machine and the machine on which the **udebug** command was run. In the first example, this is **0**.

  Note: If the message ******clock may be bad** appears, the difference between the two clocks is greater than 40 seconds, and you must synchronize the clocks on all of the server machines in the RPC server group.

- The IP address of the server machine that this server voted for to be the synchronization site and the time that the vote was cast. In the first example, this is **192.56.207.26** at **-10**.

  Note: Unless noted otherwise, all time is calculated and displayed as the number of seconds before (negative) or after (positive) the current time according to the clock on the local machine on which the **udebug** command is run.
- The time at which the last round of sync-site voting began. In the first example, this is **−11**.
- The version of the database in use on this server machine. In the first example, this is **750478963.1**.
- Whether the server is the synchronization site; if it is, the duration of the synchronization site status and the number of servers in the RPC server group are also provided. In the first example, the message **I am not sync site** indicates that the server is not the synchronization site.
- If the server is *not* the synchronization site, the following information is displayed:
  - The IP address of the lowest server in the RPC server group and the time that a beacon was last sent from that server to the specified server. In the first example, this is **192.56.207.26** at **-10**.
  - The IP address of the synchronization site and the time that a beacon was last sent from that server. In the first example, this is **192.56.207.26** at **-10**.

  If the server is the synchronization site, the current state of the server is displayed, using one of the following flags. In the second example, this is **1f**.
  - **1** − Indicates that the server is the synchronization site.
  - **3** − Indicates that the server is the synchronization site and that it has found the latest version of the database.
  - **7** − Indicates that the server is the synchronization site and that it has fetched the latest version of the database.
  - **f** − Indicates that the server is the synchronization site and a quorum has been reached in the RPC server group, but the synchronization site has not distributed the latest version of the database to all servers in the RPC server group.
  - **1f** − Indicates that server is the synchronization site, a quorum has been reached in the RPC server group, and the synchronization site has distributed the latest version of the database to all servers in the RPC server group.
- The version of the database in use at the synchronization site. In the first example, this is **750478963.1**.
- The total number of database pages locked and the number of database pages locked for write purposes on the server. (Anything other than a 0 indicates database activity.) In the first example, this is **0** and **0**.
- The time that the server was the synchronization site, if it ever has been, or a message indicating that the server has never been the synchronization site. In the first example, the message **This server has never been sync site** indicates that the server has never been the synchronization site.

If the **udebug** command specifies the synchronization site of the RPC server group or if the **-long** option is used with the command, the following additional information is displayed for each of the other database servers in the RPC server group:

- The IP address of each server machine. In the second example, the first server machine listed has the IP address **192.56.207.36**.

- The version of the database in use on each server machine. (A value of **0.0** indicates that the server does not have a version of the database.) In the second example, the first server listed uses the database version **750478963.1**.
- The last time a vote was received from this server by the server specified with the **-server** option. In the second example, the server with IP address **192.56.207.26** received a vote from the first server with IP address **192.56.207.36** at **-8**.
- The last time a beacon requesting a vote was sent to each server. In the second example, the first server received a beacon at **-9**.
- The last vote, yes or no, cast by each server. In the second example, the first server cast a **yes** vote.
- A flag (**dbcurrent**) indicating whether the version of the database in use on each server machine is current with the synchronization site; 0 indicates no, 1 indicates yes. In the second example, the first server has a current version of the database.
- A flag (**up**) indicating whether the corresponding server process on each server machine is up; 0 indicates no, 1 indicates yes. In the second example, the first server is up.
- A flag (**beaconSince**) indicating whether a response (vote) to the latest beacon was sent by each server to the synchronization site. In the second example, the first server sent a response to the latest beacon.

EXAMPLES

The following command displays information on a specified database server that is not a synchronization site:

```
$ udebug /.:/fs fs2
Host 192.56.207.146, his time is 0
Vote: Last yes vote for 192.56.207.26 at -10 (sync site); Last vote started
 at -11
Local db version is 750478963.1
I am not sync site
Lowest host 192.56.207.26 at -10
Sync host 192.56.207.26 at -10
Sync site's db version is 750478963.1
0 locked pages, 0 of them for write
This server has never been sync site
```

The following command displays information on a specified database server that is a synchronization site; the output also provides information on the other database servers in the RPC server group:

```
$ udebug /.:/fs fs4


Host 192.56.207.26, his time is 0
Vote: Last yes vote for 192.56.207.26 at -9 (sync site); Last vote started
at -9
Local db version is 750478963.1
I am sync site until 81 (4 servers)
Recovery state 1f
Sync site's db version is 750478963.1
0 locked pages, 0 of them for write
This server last became sync site at -38195

Server 192.56.207.36: (db 750478963.1)
    last vote rcvd at -8, last beacon sent at -9, last vote was yes
    dbcurrent=1, up=1 beaconSince=1
```

```
Server 192.56.207.146: (db 750478963.1)
    last vote rcvd at -8, last beacon sent at -9, last vote was yes
    dbcurrent=1, up=1 beaconSince=1
Server 192.56.207.94: (db 750478963.1)
    last vote rcvd at -8, last beacon sent at -9, last vote was yes
    dbcurrent=1, up=1 beaconSince=1
```

RELATED INFORMATION

Commands: **bakserver(8dfs)**, **flserver(8dfs)**.

upclient

Purpose

Initializes the client portion of the Update Server

Synopsis

upclient -server *machine***-path** { *filename* | *directory_name*}[**-time** *frequency*]
[**-file** *log_file*]**[-verbose][-help]**

OPTIONS

-server *machine*
Specifies the DCE pathname of the machine (for example, */.../abc.com/hosts/fs1*) whose files are to be periodically checked. The machine should be either the System Control machine for the cell or domain or the Binary Distribution machine for the local machine's CPU/operating system type.

-path { *filename* **|** *directory_name***}**
Names each file or directory to be checked periodically on the local disk of the machine specified with the **-server** option. If multiple paths are supplied, they must be unique, disjoint trees in the file system. Paths are examined from left to right; paths that intersect with previous paths used in the command are logged as errors (if a log file is specified with the **-log** option) and ignored.

If you specify a directory, the **upclient** process recursively checks all files and directories located beneath the specified directory. Therefore, you can specify a **/** (slash) to check all files and directories on the local disk of the machine specified with the **-server** option.

-time *frequency*
Specifies in number of seconds how often the **upclient** process is to check each file or directory specified with the **-path** option. The default is 300 seconds (5 minutes).

-file *log_file*
Names the log file on the local machine to which errors are to be written. Because multiple **upclient** processes can be run on one machine, choose a distinct filename for the log. If this option is omitted, no errors are logged.

-verbose
Directs the **upclient** process to produce detailed information about its actions each time it checks for new versions of files (as specified with the **-time** option). The process lists each file and directory object it checks and any changes it makes to local versions of these objects. The output is sent to standard error.

-help Prints the online help for this command. All other options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **upclient** command. See the **bos help** and **bos apropos** reference pages for examples of using these commands.

DESCRIPTION

The **upclient** command initializes the client portion (**upclient** process) of the Update Server. The **upclient** process periodically checks specified files and directories on the local disk of the server machine specified with the **-server** option to be sure they match the corresponding files and directories on the local machine (the machine running the **upclient** process). If a file on the specified server machine does not match the version on the local machine, the **upclient** process requests the newer version from the server portion (**upserver** process) of the Update Server on the specified machine. It then overwrites the local version of the file with the newer version.

The **upclient** process is usually started and controlled by the BOS Server; if it is not, execute the **upclient** process as a background process. The binary file for the **upclient** process resides in *dceloca*l**/bin/upclient**.

The **-time** option specifies how often the **upclient** process is to check for changed versions of files and directories. The **-path** option specifies the files and directories the **upclient** process is to check. If you specify a directory, the **upclient** process recursively checks all files and directories located beneath the specified directory. To check all files and directories on the indicated server machine, specify a **/** (slash) with the **-path** option.

If you specify multiple files and directories with the **-path** option, the paths must be disjoint (nonintersecting). Pathnames specified with the **-path** option are examined from left to right. Any path that intersects with a previous path is logged as an error (if a log file is named with the **-file** option) and ignored. An error also occurs if the **-path** option names a file or directory that the **upserver** process on the specified server machine is not directed to distribute. Because multiple **upclient** processes can be run on a single machine, a filename specified with the **-file** option must be distinct.

Finally, the machine running the **upclient** process must be named in the **admin.up** file on the machine running the **upserver** process (the machine specified with the **-server** option). Otherwise, the local machine's **upclient** process is not permitted to access files from the **upserver** process.

Privilege Required

You must be logged in as **root** on the local machine.

EXAMPLES

The following command starts the **upclient** process running on the local machine. The process is to check every 180 seconds (3 minutes) for changes to the binary files in the directory **/rs_aix41/bin** on the Binary Distribution machine named */.../abc.com/hosts/fs1*. Errors are written to the file named **/tmp/fs1/UpclientLog** on the local disk of the machine running the **upclient** process.

```
$ upclient -s /.../abc.com/hosts/fs1 -p /rs_aix41/bin -t 180 -l /tmp/fs1/UpclientLog
```

RELATED INFORMATION

Commands: **upserver(8dfs)**.

Files: **admin.up(4dfs)**.

upserver

Purpose

Initializes the server portion of the Update Server

Synopsis

upserver -path { *filename* | *directory_name*}[**-adminlist** *filename*][**-help**]

OPTIONS

-path { *filename* **|** *directory_name***}**
Names each file or directory to be distributed (exported) in unencrypted form upon request. If multiple paths are supplied, they must be unique, disjoint trees in the file system. Paths are examined from left to right; paths that intersect with previous paths used in the command are logged as errors and ignored.

All files and subdirectories located beneath a specified directory can be distributed from the local machine. Therefore, you can specify a **/** (slash) to allow all files and directories on the local disk of the machine to be distributed.

-adminlist *filename*
Specifies the file that contains server principals authorized to request files from the local machine. If you do not specify the complete pathname of a file, the file is assumed to reside in the current working directory. If this option is omitted, the **upserver** process uses the default file (*dcelocal***/var/dfs/admin.up**).

-help Prints the online help for this command. All other options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **upserver** command. See the **bos help** and **bos apropos** reference pages for examples of using these commands.

DESCRIPTION

The **upserver** command initializes the server portion (**upserver** process) of the Update Server. The **upserver** process distributes files from the local disk of a machine in response to requests from the client portion (**upclient** process) of the Update Server running on other machines. An **upserver** process should be run on the System Control machine for the cell or domain and on the Binary Distribution machine for each CPU/operating system type.

The **upserver** process is usually started and controlled by the BOS Server; if it is not, execute the **upserver** process as a background process. The binary file for the **upserver** process resides in *dcelocal***/bin/upserver**.

The **-path** option specifies the files and directories the **upserver** process can distribute from the local disk of the machine on which it is run. The **upserver** process can distribute all files and subdirectories located beneath a specified directory on the local machine; an **upclient** process can request and receive any file from the specified directory. Specify a **/** (slash) to allow all files and directories on the local disk of the machine to be distributed.

If the **-path** option names only a single file from a directory, an **upclient** process can request and receive only that file. An **upclient** process that requests the entire directory in which the file resides receives no files. If you specify multiple files and directories, the paths must be disjoint (nonintersecting). Paths are examined from left to right; any path that intersects with a previous path is logged as an error and ignored.

The **upserver** process writes error messages to the *dcelocal***/var/dfs/adm/UpLog** event log file. When the **upserver** process is started, it creates the **UpLog** file if the file does not already exist. It then appends messages to the file. If the file exists when the **upserver** process is started, the process moves it to the **UpLog.old** file in the same directory (overwriting the current **UpLog.old** file if it exists) before creating a new version to which to append messages.

Only one **upserver** process should be run on a machine at one time. The **upserver** process automatically creates the *dcelocal***/var/dfs/admin.up** file if the file does not already exist. A machine must be named in the **admin.up** file for its **upclient** process to be permitted to access files from the **upserver** process.

Privilege Required

You must be logged in as **root** on the local machine.

EXAMPLES

The following command specifies that files from the directories **/rs_aix41/bin** and */usr/mike*, which reside on the local disk of the machine, are to be exported upon request from **upclient** processes. The indicated paths are nonintersecting, so the command executes as intended.

```
$ upserver -path /rs_aix41/bin /usr/mike
```

The following command specifies that files from the directories **/rs_aix41/bin**, */usr/mike/public*, and */usr/mike*, which are located on the local disk, are to be exported upon request. However, because the path */usr/mike/public* is a subset of the path */usr/mike*, the command logs an error in the **UpLog** file and ignores the */usr/mike* path. The */usr/mike/public* path is exported as requested.

```
$ upserver -path /rs_aix41/bin /usr/mike/public /usr/mike
```

Had */usr/mike* been specified before */usr/mike/public* in the previous command, the */usr/mike/public* path would have been logged as an error in the **UpLog** file and ignored. In this case, the */usr/mike* path would have been exported as intended.

RELATED INFORMATION

Commands: **upclient(8dfs)**.

Files: **admin.up(4dfs)**, **UpLog(4dfs)**.

Appendix. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make them available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
IBM

AFS and DFS are trademarks of Transarc Corporation, in the United States, or other countries, or both.

Open Software Foundation, OSF, the OSF logo, OSF/1, OSF/Motif, and Motif are registered trademarks of the Open Software Foundation, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Index

Special Characters

A

B

C

D

E

F

G

Q

R

S

T

U

V

W

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.