DCE 1.2.2 DFS Administration Guide and Reference

OSF® DCE Product Documentation

The Open Group

OSF® DCE Product Documentation:

DCE 1.2.2 DFS Administration Guide and Reference, (Volume 1)
ISBN 1–85912–123–3
Document Number F209A

DCE 1.2.2 DFS Administration Guide and Reference, (Volume 2)
ISBN 1–85912–128–4
Document Number F209B

Published in the U.K. by The Open Group, 1997.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:
OGPubs@opengroup.org

OTHER NOTICES

Contents

Part 2. DCE 1.2.2 DFS Administration Reference

List of Figures

List of Tables

Preface

The Open Group

The Open Group is the leading vendor-neutral, international consortium for buyers and suppliers of technology. Its mission is to cause the development of a viable global information infrastructure that is ubiquitous, trusted, reliable, and as easy-to-use as the telephone. The essential functionality embedded in this infrastructure is what we term the IT DialTone. The Open Group creates an environment where all elements involved in technology development can cooperate to deliver less costly and more flexible IT solutions.

Formed in 1996 by the merger of the X/Open Company Ltd. (founded in 1984) and the Open Software Foundation (founded in 1988), The Open Group is supported by most of the world's largest user organizations, information systems vendors, and software suppliers. By combining the strengths of open systems specifications and a proven branding scheme with collaborative technology development and advanced research, The Open Group is well positioned to meet its new mission, as well as to assist user organizations, vendors, and suppliers in the development and implementation of products supporting the adoption and proliferation of systems which conform to standard specifications.

With more than 200 member companies, The Open Group helps the IT industry to advance technologically while managing the change caused by innovation. It does this by:

- consolidating, prioritizing, and communicating customer requirements to vendors
- conducting research and development with industry, academia, and government agencies to deliver innovation and economy through projects associated with its Research Institute
- managing cost-effective development efforts that accelerate consistent multi-vendor deployment of technology in response to customer requirements
- adopting, integrating, and publishing industry standard specifications that provide an essential set of blueprints for building open information systems and integrating new technology as it becomes available
- licensing and promoting the Open Brand, represented by the "X" mark, that designates vendor products which conform to Open Group Product Standards
- promoting the benefits of IT DialTone to customers, vendors, and the public.

The Open Group operates in all phases of the open systems technology lifecycle including innovation, market adoption, product development, and proliferation. Presently, it focuses on seven strategic areas: open systems application platform development, architecture, distributed systems management, interoperability, distributed computing environment, security, and the information superhighway. The Open Group is also responsible for the management of the UNIX trademark on behalf of the industry.

The Development of Product Standards

This process includes the identification of requirements for open systems and, now, the IT DialTone, development of CAE and Preliminary Specifications through an industry consensus review and adoption procedure (in parallel with formal standards work), and the development of tests and conformance criteria.

This leads to the preparation of a Product Standard which is the name used for the documentation that records the conformance requirements (and other information) to which a vendor may register a product. There are currently two forms of Product

Standard, namely the Profile Definition and the Component Definition, although these will eventually be merged into one.

The “X” mark is used by vendors to demonstrate that their products conform to the relevant Product Standard. By use of the Open Brand they guarantee, through the X/Open Trade Mark License Agreement (TMLA), to maintain their products in conformance with the Product Standard so that the product works, will continue to work, and that any problems will be fixed by the vendor.

Open Group Publications

The Open Group publishes a wide range of technical documentation, the main part of which is focused on specification development and product documentation, but which also includes Guides, Snapshots, Technical Studies, Branding and Testing documentation, industry surveys, and business titles.

There are several types of specification:

CAE Specifications

CAE (Common Applications Environment) Specifications are the stable specifications that form the basis for our Product Standards, which are used to develop X/Open branded systems. These specifications are intended to be used widely within the industry for product development and procurement purposes.

Anyone developing products that implement a CAE Specification can enjoy the benefits of a single, widely supported industry standard. Where appropriate, they can demonstrate product compliance through the Open Brand. CAE Specifications are published as soon as they are developed, so enabling vendors to proceed with development of conformant products without delay.

Preliminary Specifications

Preliminary Specifications usually address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. They are published for the purpose of validation through implementation of products. A Preliminary Specification is not a draft specification; rather, it is as

stable as can be achieved, through applying The Open Group's rigorous development and review procedures.

Preliminary Specifications are analogous to the trial-use standards issued by formal standards organizations, and developers are encouraged to develop products on the basis of them. However, experience through implementation work may result in significant (possibly upwardly incompatible) changes before its progression to becoming a CAE Specification. While the intent is to progress Preliminary Specifications to corresponding CAE Specifications, the ability to do so depends on consensus among Open Group members.

Consortium and Technology Specifications

The Open Group publishes specifications on behalf of industry consortia. For example, it publishes the NMF SPIRIT procurement specifications on behalf of the Network Management Forum. It also publishes Technology Specifications relating to OSF/1, DCE, OSF/Motif, and CDE.

Technology Specifications (formerly AES Specifications) are often candidates for consensus review, and may be adopted as CAE Specifications, in which case the relevant Technology Specification is superseded by a CAE Specification.

In addition, The Open Group publishes:

Product Documentation

This includes product documentation—programmer's guides, user manuals, and so on—relating to the Prestructured Technology Projects (PSTs), such as DCE and CDE. It also includes the Single UNIX Documentation, designed for use as common product documentation for the whole industry.

Guides

These provide information that is useful in the evaluation, procurement, development, or management of open systems, particularly those that relate to the CAE Specifications. The Open Group Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming conformance to a Product Standard.

Technical Studies

Technical Studies present results of analyses performed on subjects of interest in areas relevant to The Open Group's Technical Program. They

are intended to communicate the findings to the outside world so as to stimulate discussion and activity in other bodies and the industry in general.

Versions and Issues of Specifications

As with all live documents, CAE Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new Version indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it replaces the previous publication.
- A new Issue indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on the World-Wide Web at *http://www.opengroup.org/public/pubs*.

Ordering Information

Full catalogue and ordering information on all Open Group publications is available on the World-Wide Web at *http://www.opengroup.org/public/pubs*.

This Book

The *DCE 1.2.2 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 OSF® DCE Revision 1.2.2 offering. 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. The *DCE 1.2.2 Release Notes* contain instructions for installing and building DCE source code, and they contain release-specific information about DFS.

Document Usage

The *DCE 1.2.2 DFS Administration Guide and Reference* is divided into the following parts:

- Volume 1
  Document Number F209A, ISBN 1–85912–123–3
  - — Part 1. DCE 1.2.2 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
- Volume 2
  Document Number F209B, ISBN 1–85912–128–4
  - — Part 2. DCE 1.2.2 DFS Administration Reference
    - Chapter 12. Configuration Files
    - Chapter 13. Administrative Commands
  - — Appendix A. The DFS/NFS Secure Gateway

Related Documents

For additional information about the Distributed Computing Environment, refer to the following documents:

- *DCE 1.2.2 Introduction to OSF DCE*
  Document Number F201, ISBN 1–85912–182–9
- *DCE 1.2.2 Command Reference*
  Document Number F212, ISBN 1–85912–138–1
- *DCE 1.2.2 Application Development Reference*
  Document Number F205A, ISBN 1–85912–103–9 (Volume 1)
  Document Number F205B, ISBN 1–85912–108–X (Volume 2)
  Document Number F205C, ISBN 1–85912–159–4 (Volume 3)
- *DCE 1.2.2 Administration Guide—Introduction*
  Document Number F207, ISBN 1–85912–113–6
- *DCE 1.2.2 Administration Guide—Core Components*
  Document Number F208, ISBN 1–85912–118–7
- *DCE 1.2.2 Application Development—Introduction and Style Guide*
  Document Number F202, ISBN 1–85912– 187–X
- *DCE 1.2.2 Application Development Guide—Core Components*
  Document Number F203A, ISBN 1–85912–192–6 (Volume 1)
  Document Number F203B, ISBN 1–85912–154–3 (Volume 2)
- *DCE 1.2.2 Application Development Guide—Directory Services*
  Document Number F204, ISBN 1–85912–197–7
- *DCE 1.2.2 GDS Administration Guide and Reference*
  Document Number F211, ISBN 1–85912–133–0
- *DCE 1.2.2 File-Access Administration Guide and Reference*
  Document Number F216, ISBN 1–85912–158–6
- *DCE 1.2.2 File-Access User's Guide*
  Document Number F217, ISBN 1–85912–163–3
- *DCE 1.2.2 Testing Guide*
  Document Number F215, ISBN 1–85912–153–5
- *DCE 1.2.2 File-Access FVT User's Guide*
  Document Number F210, ISBN 1–85912–189–6

- *DCE 1.2.2 Release Notes*
  Document Number F218, ISBN 1–85912–168–3

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.

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.

Problem Reporting

If you have any problems with the software or vendor-supplied documentation, contact your software vendor's customer service department. Comments relating to this Open Group document, however, should be sent to the addresses provided on the copyright page.

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 *DCE 1.2.2 Administration Guide—Introduction* and *DCE 1.2.2 Testing Guide*.

Trademarks

Part 1

DCE 1.2.2 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 Part 1 of this guide and reference.

1.1 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.

1.1.1 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 for more information about the roles of DFS server machines.)

1.1.2 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.

1.1.3 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.

1.1.4 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.

1.1.5 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.

1.1.6 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. 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.)

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.

1.1.7 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.

1.1.8 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 both DCE LFS filesets and non-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.

1.1.9 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.

1.1.10 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.

1.1.11 Access to DFS from NFS

The DFS/NFS Secure Gateway provides authenticated access to DFS via the Network File System (NFS). The DFS/NFS Secure 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 DFS/NFS Secure Gateway, users of NFS clients have only unauthenticated access to data in the DFS filespace.

To use the DFS/NFS Secure 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; authenticating to DCE provides these users the privileges and permissions associated with their DCE identities. Mounting the DCE namespace also provides unauthenticated access to DFS to users who do not have DCE accounts. (See Appendix A for information about configuring and using the DFS/NFS Secure Gateway.)

1.2 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.

1.2.1 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.

1.2.2 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*.

1.2.3 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.

1.2.4 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.

1.2.5 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.

1.2.6 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.

1.2.7 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.

1.2.8 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.

1.2.9 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.

1.2.10 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.

1.3 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 *DCE 1.2.2 Introduction to OSF DCE* and understands the following DCE components:

- The DCE Security Service, especially the keytab file and ACLs. (See the *DCE 1.2.2 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 *DCE 1.2.2 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 *DCE 1.2.2 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 *DCE 1.2.2 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 *DCE 1.2.2 Introduction to OSF DCE* and the *DCE 1.2.2 Administration Guide—Core Components* for more information on these components and their mutual interaction.)

1.3.1 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 **secd** process runs on all DFS server machines to provide access to the security services in the previous list.

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.

1.3.2 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 Service (GDS) supports the global naming environment between cells and outside of cells. GDS is an implementation of a directory services standard known as X.500.
- 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.

1.3.2.1 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
```

1.3.2.2 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. (See your vendor's installation and configuration documentation for information about specifying a name other than **fs** as the junction for a cell's DFS filespace.)

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.

1.3.3 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.

1.3.4 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.

1.4 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 configure the Cache Manager, 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.

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. DFS also includes an additional command suite, **dfsgw**, that is used with the DFS/NFS Secure Gateway 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). Section 1.5 provides information about the structure of DFS commands and describes how to receive online help for them.

Refer to Part 2 of this guide and reference for detailed discussions of the various DFS commands. Refer to the *DCE 1.2.2 Command 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.

1.4.1 Fileset Management Commands

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

1.4.1.1 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.

1.4.1.2 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.

1.4.2 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.

1.4.2.1 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.

1.4.2.2 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.

1.4.2.3 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.

1.4.3 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.

1.4.4 DFS/NFS Secure Gateway Commands

Commands available with the DFS/NFS Secure Gateway provide authenticated access to data in DFS from an NFS client. You can use the commands to perform the following tasks:

- Obtain DCE credentials from an NFS client with the **dfs_login** command.
- Destroy DCE credentials obtained from an NFS client with the **dfs_logout** command.

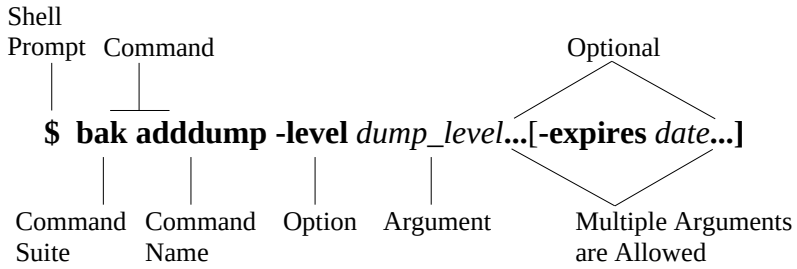
- Administer DCE credentials for NFS users from a Gateway Server with the **dfsgw** commands.

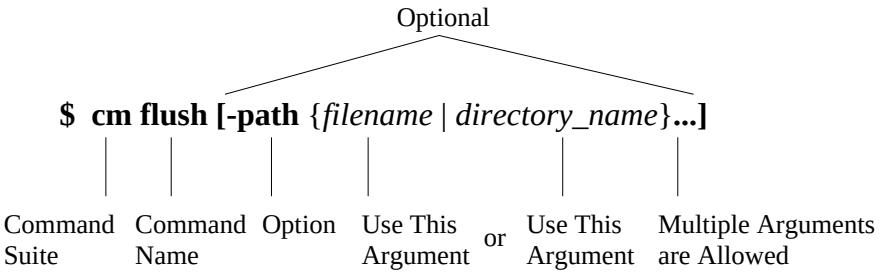
1.5 DFS Command Structure and Help

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).

1.5.1 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 Part 2 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:

- Complete command:

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

- Abbreviated command name and abbreviated option:

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

- Abbreviated command name and omitted option:

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

Note: The **dfs_login** and **dfs_logout** commands provided with the DFS/NFS Secure Gateway do not provide the shortcuts and help available with other DFS commands. See the reference pages for these commands for information about using them.

1.5.2 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 **man** followed by the command suite.

```
$ man command_suite
```

- To view the reference page for an individual 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 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.

2.1 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. (See your vendor's installation and configuration documentation for more information about fulfilling these requirements.)

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 provides detailed information about the procedures used to create and maintain administrative lists. (See Part 2 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.

2.1.1 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, both machines act as Fileset Database machines and Backup Database machines to replicate the databases. For each database, one of the machines automatically assumes the role of the synchronization site

and houses the source copy of the database. If one of the machines becomes unavailable, the information in the database may not be able to be changed. (See Section 2.5 for a detailed description of database synchronization.)

- 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.

2.1.1.1 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 Section 2.1.1.2 for a description of the Binary Distribution machine.)
- A BOS Server (**bosserver** process). (See Section 2.1 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.

2.1.1.2 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 Section 2.1 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.

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.

2.1.1.3 File Server Machines

A File Server machine is used to store and export DCE LFS or non-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 Section 2.1 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 non-LFS 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 Section 2.4 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 Part 2 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 Section 2.2.2. (See Part 2 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 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 Section 2.1.1.7 for more information about configuring a client machine to export data.)

2.1.1.4 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 Section 2.1 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 for more information about creating server entries in the FLDB.)

2.1.1.5 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 Section 2.1 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 for more information on configuring and using Tape Coordinator machines.)

2.1.1.6 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 Section 2.2.2 for more information about the Cache Manager and the **dfsd** and **dfsbind** processes.)

2.1.1.7 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 Section 2.1.1.3 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.

2.1.2 Summary of DFS Machine Roles

Table 2-1 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 Section 2.1.1 for expanded descriptions of the machine roles.)

Note: Table 2-1 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 2–1. 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.
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.

Machine Role	Purpose	Processes	Suggestions
Backup Database machine	To store the Backup Database	**bosserver** **bakserver** **upclient**[1] **upclient**[2]	Configure three Backup Database machines. Configure Backup Database machines as Fileset Database machines.
DFS client machine	To serve as a single-user or multiuser workstation; to access files for application programs	**dfsd** **dfsbind**	Export DCE LFS and non-LFS data from the machine by running **bosserver**, **ftserver**, **fxd**, **upclient**[2], and optionally **repserver**, creating an RPC binding in CDS, registering a DFS server principal in the Registry Database, and creating a server entry in the FLDB.

2.2 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.

2.2.1 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. (See your vendor's documentation for information about the files that reside on the local disk of a server machine.)

- 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.

2.2.2 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 Part 2 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. (See your vendor's documentation for information about the files that reside on the local disk of a client machine.)

- The *dcelocal*/**bin/dfsbind** command is the start-up command for the **dfsbind** process.
- The *dcelocal*/**bin/dfsd** command is the start-up command for the Cache Manager.
- The *dcelocal*/**etc/CacheInfo** file is a file that specifies aspects of Cache Manager configuration.
- The *dcelocal*/**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.

2.2.3 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.

2.2.3.1 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 9.

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.

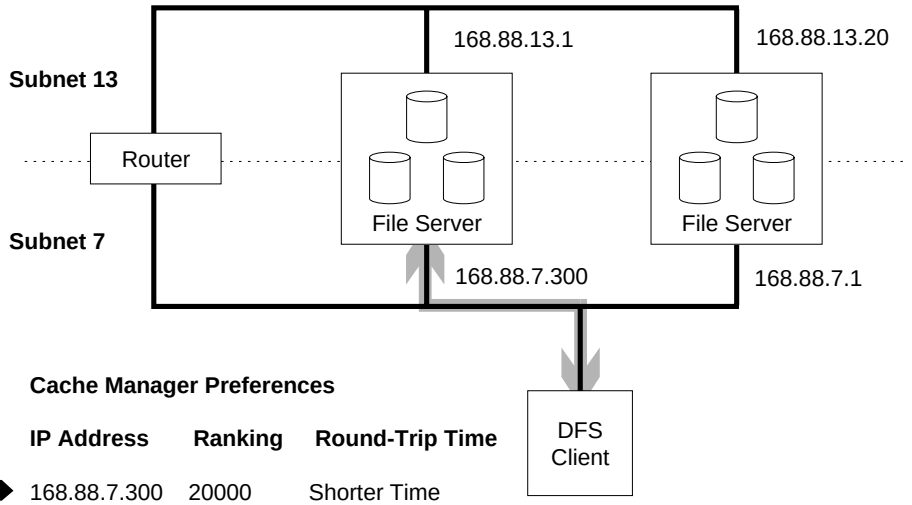
You can change Cache Manager preferences by using the **cm setpreferences** command. Additionally, you can create a file specifying server preferences that is read each time a Cache Manager is initialized, thus providing a method for overriding the default server preference values. You can also load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

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.

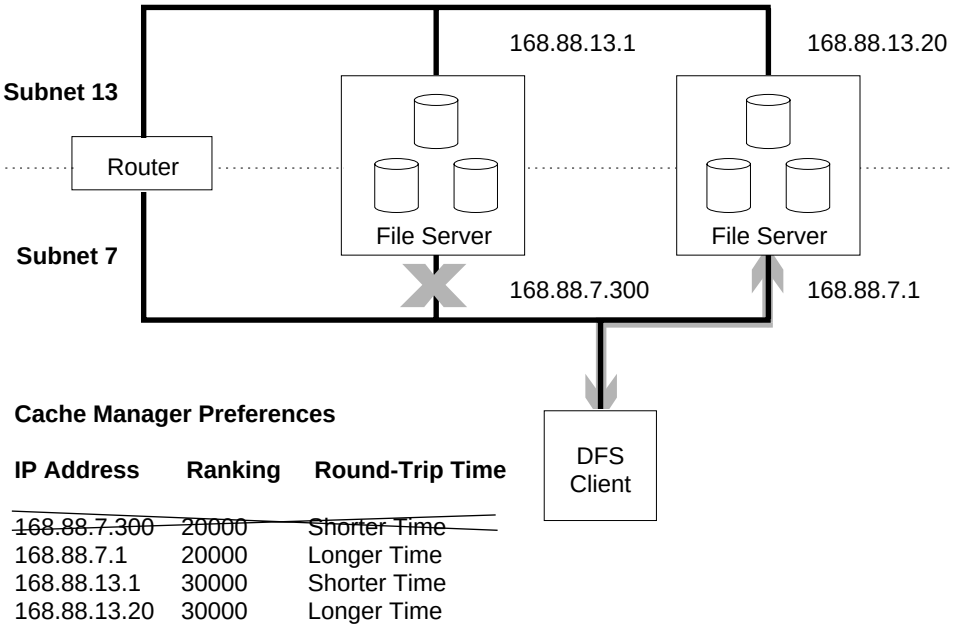
The following simplified scenario illustrates how multihomed servers can be configured to make the most efficient use of the local network. In this example, a read-only fileset is replicated on two File Servers. The File Servers have connections to both subnetworks within the network, and these connections 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 2-1.

Figure 2–1. Cache Manager Contacting File Server Address With Lowest Rank



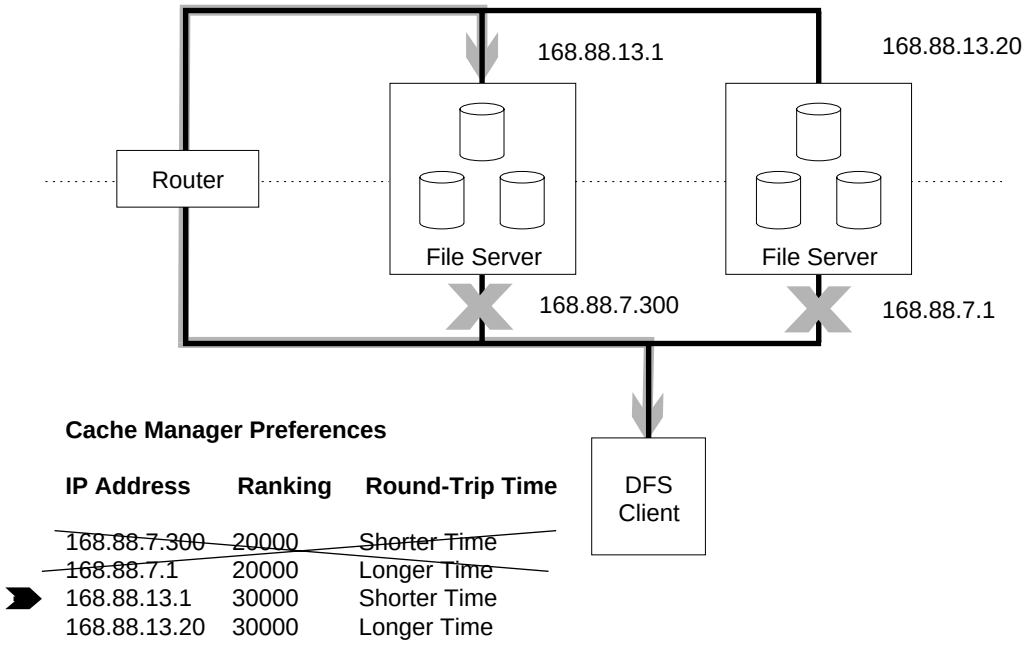
Should the Cache Manager lose contact with the preferred File Server connection (either through a network or server problem), the Cache Manager again consults its preference list and attempts to contact a suitable File Server at the address with the next lowest rank, as shown in Figure 2-2. In this figure, 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 2–2. 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 2-3. 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 2–3. 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.

2.2.3.2 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 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.

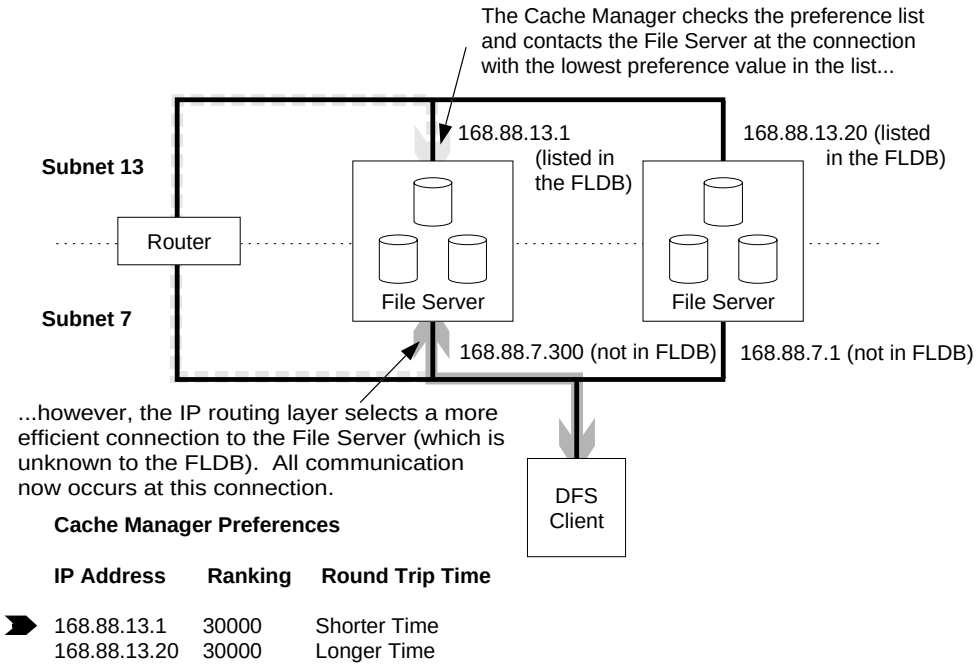
  Client preference lists are transient in that they are reestablished at their default values each time the Cache Manager is initialized. However, a list of preferences can be loaded into the Cache Manager at initialization through a preferences file. Chapter 9 explains how to both create such a file and ensure that it is loaded each time the Cache Manager is initialized.

2.2.3.3 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 2-4.

Figure 2–4. 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.

2.2.3.4 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.

2.3 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 for complete information about creating and mounting filesets; see Part 2 of this guide and reference for complete information about **fts** and other DFS commands.)

2.3.1 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 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.

2.3.2 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 for more information on additional rules for naming filesets.)

2.3.3 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_aix32**) 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 Section 2.3.7 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, which runs the OSF/1 and AIX 3.2 operating systems, can configure part of their file tree as shown in Table 2-2.

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

Fileset Name	Mount Point
pmax_osf1	**/.../abc.com/fs/pmax_osf1**
pmax_osf1.bin	**/.../abc.com/fs/pmax_osf1/bin**
pmax_osf1.etc	**/.../abc.com/fs/pmax_osf1/etc**
rs_aix32	**/.../abc.com/fs/rs_aix32**
rs_aix32.bin	**/.../abc.com/fs/rs_aix32/bin**
rs_aix32.etc	**/.../abc.com/fs/rs_aix32/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.)

2.3.4 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**. Table 2-3 lists the organization and names of the filesets in this example.

Table 2–3. 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**

2.3.5 Moving Data from Non-LFS Directories to DCE LFS Directories

The guidelines in Section 2.3.4 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 Section 2.3.4. 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.

2.3.6 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 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 for more information about the Cache Manager and File Server machine preferences.)

2.3.7 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.

2.3.7.1 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 OSF/1:

```
$ cm sysname

Current sysname is 'pmax_osf1'

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

/.../abc.com/fs/pmax_osf1
```

On a machine running AIX 3.2:

```
$ cm sysname

Current sysname is 'rs_aix32'
```

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

```
/.../abc.com/fs/rs_aix32
```

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 OSF/1:

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

```
lrwxrwxrwx 1 root 34 Nov 22 1991 /usr/local ->
/.../abc.com/fs/@sys/usr/local
```

```
$ cd /usr/local
$ pwd
```

```
/.../abc.com/fs/pmax_osf1/usr/local
```

On a machine running AIX 3.2:

```
$ 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_aix32/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 Part 2 of this guide and reference for more information about the **cm sysname** command.)

2.3.7.2 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

2.4 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.

2.4.1 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.

2.4.2 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.

2.4.3 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 Part 2 of this guide and reference for complete information about the **fxd** command and its options.)

2.5 Data Access Security in DFS

DFS includes administrative commands to establish and modify RPC authentication levels for communications between Cache Managers and File Servers. DFS provides very flexible tools 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.

The default values for security settings at the Cache Manager and File Server ensure that communications between a Cache Manager and File Server are authenticated at the DCE packet integrity security level. All data received has been authenticated as originating at the expected host and has been verified to have not been modified during transmission. However, you can choose to set higher or lower RPC authentication levels for each Cache Manager and File Server. Note that 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 that Cache Managers in versions of DFS earlier than 1.2.2 cannot negotiate RPC authentication levels. Setting the minimum authentication level bound at a File Exporter higher than packet integrity 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 Part 2 of this guide and reference.

2.5.1 Fileset Advisory RPC Authentication Bounds

You can establish advisory minimum and maximum RPC authentication bounds for each fileset. As with the File Server RPC authentication bounds, the FLDB holds a pair of bounds for each fileset. One set of bounds governs communications with Cache Managers that are in the same cell as the File Server within which the fileset resides;

the other set of bounds controls communications with Cache Managers in foreign cells. While these advisory bounds are not currently enforced (although they may be in a future release of DFS), they do serve to bias the initial RPC authentication level when a Cache Manager attempts to access that fileset. The advisory bounds work as follows.

When the Cache Manager contacts a Fileset Location (FL) Server to ascertain the location (or locations) of a given fileset, the information returned by the FL Server includes that fileset's lower and upper RPC bounds. The Cache Manager compares its initial RPC authentication level to the range set by the advisory bounds. If the initial level falls within that range, the Cache Manager begins negotiations with a File Server using the initial level. However, if the initial level is above or below the range, the Cache Manager adjusts its initial level to match the closest bound level. (If the File Server requests that the Cache Manager lower its authentication level below the minimum level specified for the Cache Manager, the Cache Manager refuses communications with that File Server.) The Cache Manager then uses the modified initial level to begin negotiations with a File Server.

You establish the fileset advisory RPC authentication level bounds by using the **fts setprotectlevels** command. You can check if a given fileset has advisory bounds and display the bound level values by using either the **fts lsfldb** command or the **fts lsft** command.

2.6 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.

2.6.1 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.

2.6.2 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. Section 2.5.3 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.

2.6.3 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, perform the steps in Sections 2.5.3.1 and 2.5.3.2 to properly configure information for Ubik. (You may be able to use the DCE installation and configuration program to modify the database servers configured in your cell. See your vendor's installation and configuration documentation for more information.)

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.

Sections 2.5.3.1 and 2.5.3.2 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 Part 2 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 for more information about **bos** commands.)

2.6.3.1 Adding a Database Server Machine

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 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.

2.6.3.2 Removing a Database Server Machine

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.

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 *DCE 1.2.2 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 for complete details about using administrative lists; see the Security Service portion of the *DCE 1.2.2 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.

3.1 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 Sections 3.1.5 and 3.1.6 for more information about this possibility; see Section 3.1.4 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.

3.1.1 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).

3.1.1.1 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 3-1 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 3–1. 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.
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 Section 3.1.5.1 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.)

3.1.1.2 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.

3.1.1.3 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 Section 3.1.2 for a description of ACL evaluation.) 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 Version 1.1, 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 Version 1.1, 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.

3.1.1.4 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)

Table 3-2 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 3–2. 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

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 Table 3-2, 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---}
```

3.1.2 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 3-1 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.

3.1.2.1 ACL Evaluation for Local Access

If your vendor has properly configured your local operating system's **mount** command (or its equivalent), 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).

3.1.3 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 *DCE 1.2.2 Administration Guide—Core Components* for complete details about using the command to set and examine an object's ACLs.)

3.1.3.1 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.)

3.1.3.2 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.

3.1.4 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 Section 3.1.5 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**.

3.1.5 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 Section 3.1.5.4 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 Sections 3.1.5.3 and 3.1.5.4 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.

3.1.5.1 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.

3.1.5.1.1 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
```

3.1.5.1.2 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.

3.1.5.2 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:

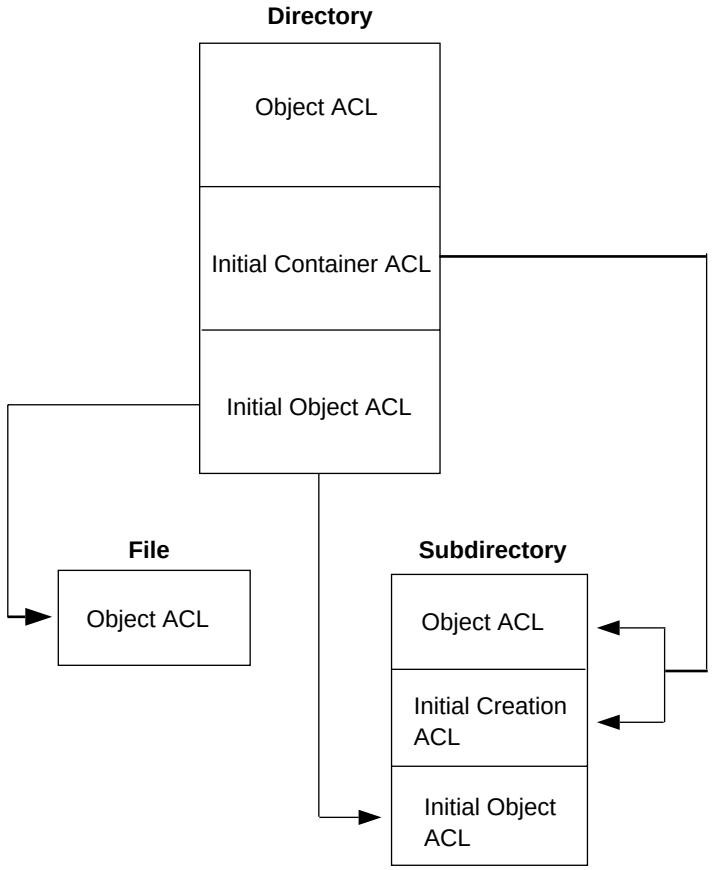
- The **r**, **w**, and **x** permissions for the file's **user_obj** entry consist of the intersection of the **user** mode bits specified when the file is created and the corresponding permissions of the **user_obj** entry of its parent directory's Initial Object Creation ACL. The **c**, **i**, and **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 3-1 illustrates ACL inheritance for files and directories.

Figure 3–1. 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.

3.1.5.3 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 Section 3.1.5.2. 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 Section 3.1.5.4 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---}
```

3.1.5.4 Mode Bits for New Objects That Do Not Inherit ACLs

The ACL inheritance algorithm described in Section 3.1.5.2 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.

3.1.5.4.1 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 Section 3.1.4). 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.

3.1.5.4.2 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.

3.1.6 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 Section 3.1.5.4.2.)

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, **user**, **group**, and **other** all receive 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 rwx-id}
{other_obj rwx-id}
```

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 Section 3.1.5.4.1.)

3.1.7 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.

3.1.8 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 3-1. 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.

3.1.8.1 ACL Entry Types for Delegation

For each ACL entry described in Table 3-1, a corresponding entry of type **_delegate** is available. Table 3-3 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_obj**, and **other_obj_delegate** entries, so these entries are omitted from the table.)

Table 3–3. 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.

Delegation Type	Key	Applies to
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 Section 3.1.1.4.)

3.1.8.2 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 Section 3.1.2. 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.

3.1.8.3 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 *DCE 1.2.2 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 *DCE 1.2.2 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 for information about administering DFS server processes.)

3.2 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 *DCE 1.2.2 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 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.

3.2.1 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 *DCE 1.2.2 Command Reference* for information about the **dcecp group** command.)

3.2.2 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.

3.2.3 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 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 non-LFS 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 non-LFS 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. Table 3-4 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 3–4. 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 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 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.)

4.1 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 Part 2 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 Section 4.2.3 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 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.

4.2 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 Part 2 of this guide and reference for more information on the **upserver** and **upclient** processes.)

4.2.1 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 Part 2 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.

4.2.2 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.

4.2.2.1 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
```

4.2.2.2 Adding Principals and Groups to Administrative Lists

To add principals and groups to 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 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.

4.2.2.3 Removing Principals and Groups from Administrative Lists

To rename principals and groups for 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.

4.2.3 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 Section 4.2.3.1.)
- 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 Section 4.3.)

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.

4.2.3.1 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 **boscron** process. (See Chapter 5 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.

4.2.3.2 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 **-authcheckingoff** 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**.

4.3 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.

4.3.1 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.
```

4.3.1.1 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**.

4.3.1.2 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 *DCE 1.2.2 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.

4.3.1.3 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**.

4.3.1.4 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**.

4.3.2 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 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 **-authcheckingon** 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
```

4.3.3 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 *DCE 1.2.2 Command 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).

5.1 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 Section 5.4 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.

5.2 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.

5.2.1 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 Part 2 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 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 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.

5.2.2 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.

5.2.2.1 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.

5.2.2.2 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.

5.3 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.

5.3.1 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**

5.3.2 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**

5.4 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.

5.4.1 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 Part 2 of this guide and reference for information about the protections the BOS Server wants to see enforced.)

Note: Your vendor can modify the default protections enforced by the BOS Server. Refer to your vendor's documentation to determine the protections that apply for your version of DFS.

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'
```

5.4.2 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'
```

5.5 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.

5.5.1 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.

5.5.2 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.

5.5.3 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***...**

5.6 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.

5.6.1 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***...**

5.6.2 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*

5.6.3 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***...**

5.7 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 Section 5.9). 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*

5.8 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 Section 5.9 for detailed information about checking and setting scheduled restart times.)

5.8.1 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**.

5.8.2 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.

5.8.3 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**.

5.8.4 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**.

5.8.5 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.

5.9 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.

5.9.1 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
```

5.9.2 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.

5.9.3 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**.

5.10 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
```

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 (**/etc/reboot** or its equivalent) for the machine to be rebooted. For example:

   # /etc/reboot

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 for information on the tasks involved in the use and maintenance of filesets.)

Note: This guide uses the term *non-LFS* to refer to *non-DCE LFS* filesets and aggregates.

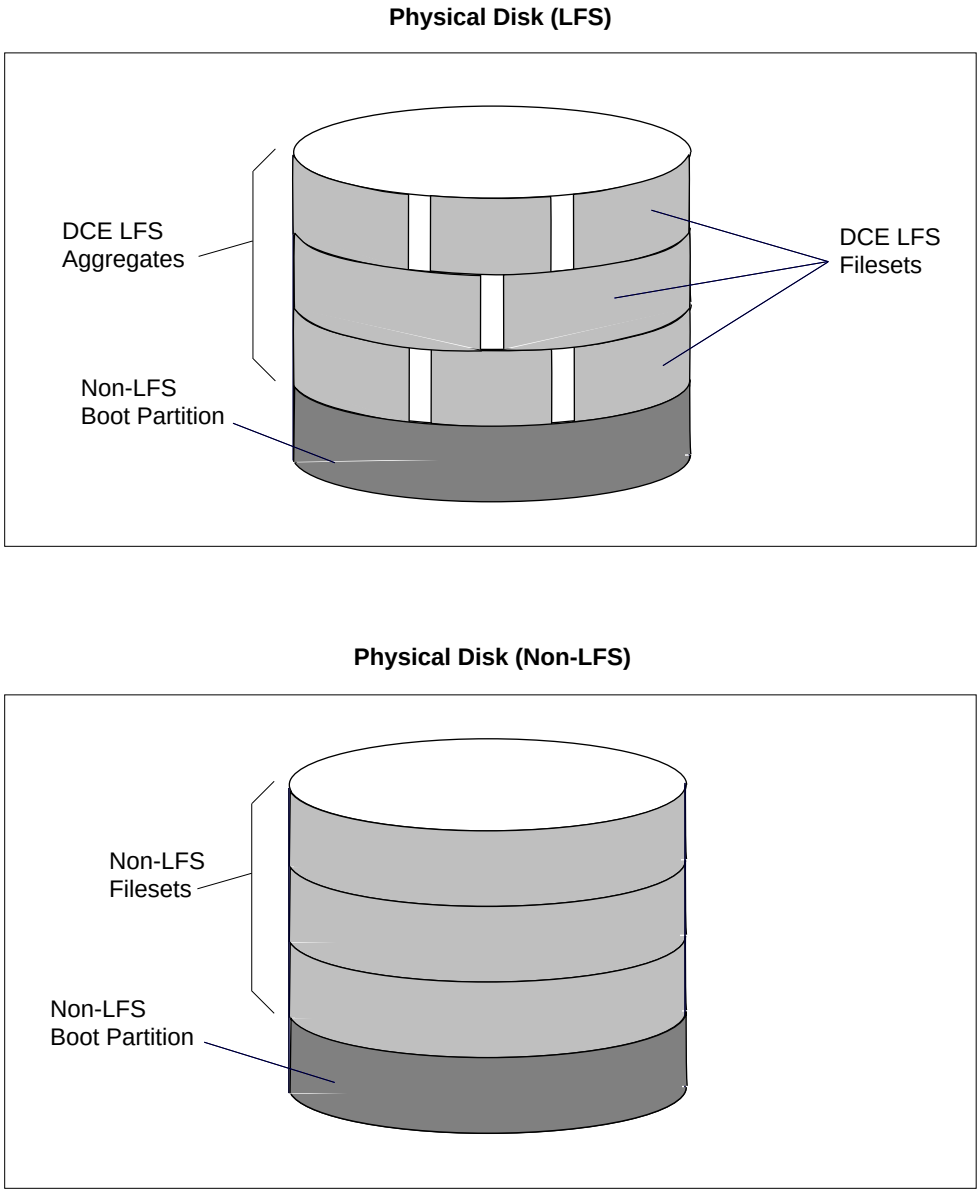
6.1 An Overview of Filesets

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 6-1 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 6–1. Comparison of DCE LFS and non-LFS Disk Partitioning Structures



6.1.1 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.

6.1.2 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.

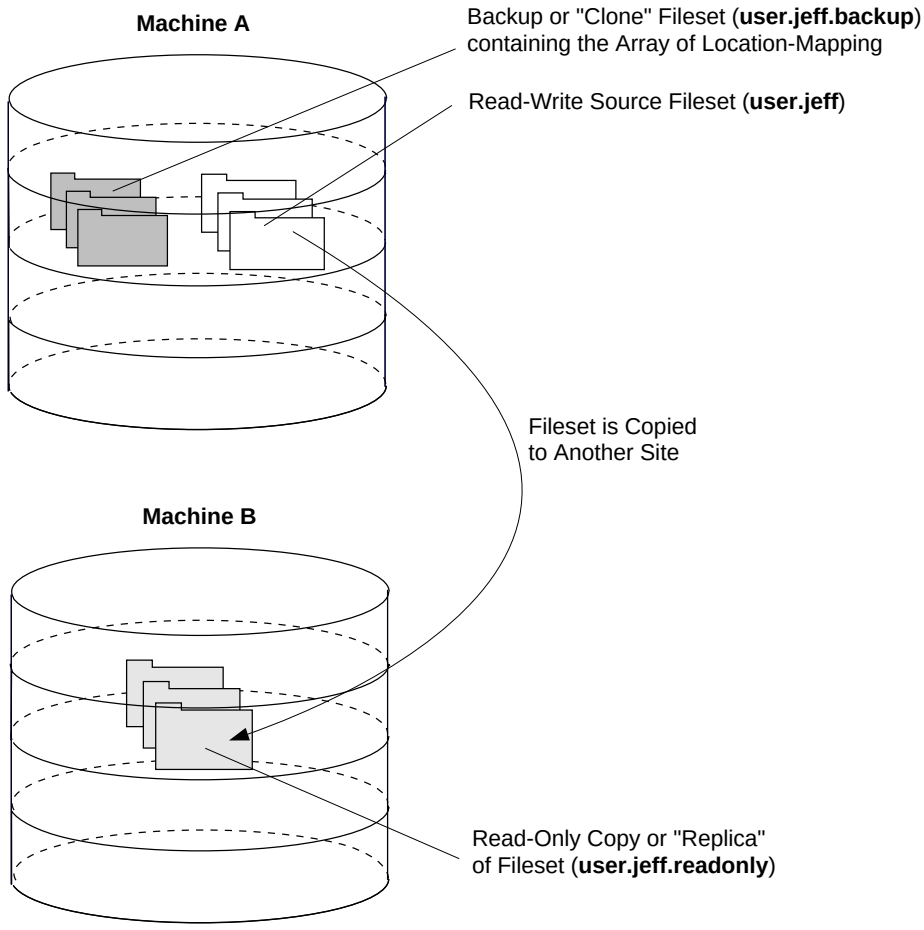
Every DCE LFS fileset has a single read/write version, which contains the modifiable versions of the files and directories in that fileset. This version is also referred to as the read/write source because the other fileset types are derived from it via replication and backup operations.

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 for more information on the backup process.)

Figure 6-2 illustrates the different types of DCE LFS filesets: read/write, read-only, and backup.

Figure 6–2. The Different Types of DCE LFS Filesets



6.1.3 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 Part 2 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 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 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.

6.1.4 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.

6.1.4.1 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_aix32.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.

6.1.4.2 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**.

6.1.5 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).

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 Section 6.2.2.3 for a description of the **dfstab** file.)

6.1.5.1 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 Section 6.4 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.

6.1.5.2 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.

6.1.6 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.

6.1.7 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 Section 6.6 for more information about the different types of mount points and how the Cache Manager accesses them.)

6.1.8 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 Part 2 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 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 Section 6.2 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.

6.2 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 Section 6.2.1 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 Part 2 of this guide and reference for more information about a specific DFS command.)

6.2.1 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 (**sec_clientd**) 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.

6.2.1.1 Creating an RPC Binding for a File Server Machine

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**.

6.2.1.2 Creating a DFS Server Principal for a File Server Machine

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 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**.

6.2.1.3 Creating a Server Entry for a File Server Machine

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.

6.2.1.3.1 Creating a Server Entry for a Machine

To create a 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 crserverentry** command to create a server entry in the FLDB for the machine:

$ fts crserverentry -server *machine* **-principal** *name* \
[-quota *entries***] [-owner** *group***]**

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.

6.2.1.3.2 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.

6.2.1.3.3 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.

6.2.1.3.4 Deleting a Server Entry for a Machine

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 the entry in the FLDB for any fileset references the server entry to be removed as the location of the fileset.

   $ 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. The command fails if a fileset entry references the server entry to be removed.

6.2.1.4 Preparing a File Server Machine for Exporting

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 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 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 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 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.

6.2.2 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.

6.2.2.1 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.

6.2.2.2 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 Part 2 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.

6.2.2.3 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.
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 Section 6.3 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 Section 6.1.4.1 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 Section 6.6 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.

6.2.3 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 Section 6.2.1 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).

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.
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 Section 6.1.4.1 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.

6.2.4 Exporting Aggregates and Partitions at System Startup

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 **rc.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. (See your vendor's DCE installation and configuration documentation for more information about installing and configuring DFS.)

6.2.5 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.

6.2.6 Using DCE LFS Filesets Locally

Note: The information in this section assumes that your vendor has properly configured your operating system's **mount** command (or its equivalent) to handle DCE LFS filesets. If this is not the case, the operations described in this section do not apply.

In addition to being exported for use in the DCE namespace, DCE LFS filesets can also be used locally, as file systems on their File Server machines. To use a DCE LFS fileset locally, use your local operating system's **mount** command (or its equivalent) to mount the fileset on the local disk of the machine.

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. Use the **mount** command for your local operating system to mount the DCE LFS fileset just as you would a non-LFS partition. The local **mount** command may be modified for DCE LFS; for example, the **mount** command may be altered to accept a fileset ID number, in which case you would specify something like the following:

```
# mount device_name directory_name fileset_ID
```

Note: If your vendor has properly configured your local operating system, you may be able to mount DCE LFS filesets automatically at system startup by including the proper information in an initialization file (**fstab** or its equivalent). Refer to your vendor's documentation for more information about mounting file systems at system startup.

6.3 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 Section 6.2.3).

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 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 Section 6.1.4.1 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 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.

6.3.1 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 for a detailed description of the **fts aggrinfo** command.)

   $ fts aggrinfo -server *machine* **-aggregate** *name*

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.

6.3.2 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.

6.4 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 Section 6.4.2.1 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 Section 6.1.3 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 for information about configuring **root.dfs** to support replication.)

6.4.1 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.

6.4.2 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 Section 6.2.1.3 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 Section 6.4.4, 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 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.

6.4.2.1 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 6–1. 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

Parameter	Default	Release Replication	Scheduled Replication
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
DefaultSiteAge (**fts setrepinfo**)	One-quarter of MaxAge	Not applicable	Optional

The system uses the guidelines listed in Table 6-1 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.)

6.4.2.1.1 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 Section 6.4.2.1.2.)

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 Section 6.4.2.1.2.)

6.4.2.1.2 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 Section 6.4.2.1.1 for details about the syntax of the **fts setrepinfo** command. See Table 6-1 for information about which of the command's options apply to the two types of replication and the dependencies between the parameters.)

6.4.2.1.3 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 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***]**

6.4.2.2 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.

6.4.2.2.1 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.

6.4.2.2.2 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.

6.4.3 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. The Replication Servers begin updating the replication sites in parallel. (This does not mean, however, that all replicas will complete their updates at the same time.)

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 Section 6.4.2 have all been performed.

6.4.3.1 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**

6.4.3.2 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.

6.4.4 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.

6.4.4.1 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.

6.4.4.2 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.

6.5 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.

6.5.1 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 Section 6.1.3 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.

6.5.2 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

6.5.3 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.

6.6 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. Section 6.6.1 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.

6.6.1 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.

6.6.1.1 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).

6.6.1.2 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.)

6.6.2 Manipulating Mount Points

The following subsections describe the commands used to create, list, and remove mount points.

6.6.2.1 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
```

6.6.2.2 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'
   ```

6.6.2.3 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 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 for introductory information about filesets.)

7.1 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 for detailed information about DCE LFS filesets, non-LFS filesets, and how the two types of filesets are created.)

7.2 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 Part 2 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 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 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.

7.3 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 Section 7.3.4 for complete instructions on using these commands with file or directory names.)

7.3.1 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>
```

7.3.2 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 Section 7.3.3.

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:

1. 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.
2. 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
```

7.3.3 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 Section 7.10 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>
```

7.3.4 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 Section 7.6, 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 for more information about **cm** commands used to configure the Cache Manager; see Part 2 of this guide and reference for more detailed information about all **cm** commands.)

7.3.4.1 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 Section 7.6 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)
```

7.3.4.2 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 Section 7.3.3 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>
```

7.3.4.3 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 Section 7.3.3 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>
```

7.3.4.4 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 Section 7.3.1 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>
```

7.3.4.5 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.
```

7.4 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.

7.4.1 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)
```

7.4.2 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)
```

7.5 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 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. Issue the **growaggr** command to increase the size of the aggregate. (See Part 2 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 disk partition 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.

7.6 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 Section 7.5).

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.

7.6.1 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.

7.6.2 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)
```

7.7 Setting Advisory RPC Authentication Bounds for Filesets

You can set upper and lower advisory RPC authentication bounds for any DCE LFS fileset. These bounds serve 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, but a future release of DFS may enforce these bounds.

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
```

7.8 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 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.

7.9 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.
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}
```

7.10 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 a non-LFS fileset or any of the three types of DCE 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 a non-LFS fileset or 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 use the **fts dump** and **fts restore** commands to dump and restore data between different types of file systems. 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 the file is restored. (See 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 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 or non-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. A non-LFS fileset must exist before the non-LFS partition on which it resides can be exported to the DCE namespace; therefore, when restoring a dump file as a non-LFS fileset, 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 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 Section 7.8 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.

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 the partition on which the fileset resides in the **dfstab** file. Note that the restored dump file overwrites all data on the non-LFS partition.

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.

7.10.1 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, **"11/22/91 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**).

7.10.2 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 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.

7.10.3 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.

7.11 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 Section 7.10.2 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.

7.11.1 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.
3. 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.

4. 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*

5. 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.

7.11.2 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.

7.11.2.1 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***]**

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.

7.11.2.2 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.

7.11.3 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.)

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.

Once you remove the fileset's FLDB entry, use the **fts delmount** command to remove the mount point for the fileset. 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.
3. 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.
4. 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.

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:

```
# dfsexport -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
```

7.12 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

7.12.1 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.

7.12.2 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***}**

7.12.3 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}
```

7.12.4 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]
```

7.13 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 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.

7.13.1 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.

7.13.2 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.

7.14 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.

7.14.1 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/rc.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.

7.14.2 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.

7.14.3 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.

7.14.4 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.

7.14.5 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.

8.1 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.

8.1.1 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.

8.1.2 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 *dcelocal*/**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 *dcelocal*/**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 *dcelocal*/**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**.

8.2 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 *dcelocal*/**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.

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.

8.3 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.
- 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
```

8.4 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.

8.4.1 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 90% 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.

8.4.2 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 Part 2 of this guide and reference for complete information about these options and the use of the **dfsd** command in general.)

8.5 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 Part 2 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 **/etc/reboot** 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.

8.6 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 90% 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 Part 2 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.

8.6.1 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
```

8.6.2 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.
```

8.6.3 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.

8.6.4 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.

8.6.5 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:

   ```
   /...:dcelocal/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.)

8.7 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.

8.7.1 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.

8.7.2 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.

8.8 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.

8.8.1 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

8.8.2 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.

8.9 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.

8.9.1 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

8.9.2 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.

8.10 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.

8.10.1 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.

8.10.2 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.

8.10.3 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

8.11 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 Section 8.12 for information about the **cm statservers** command.)

8.11.1 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
```

8.11.2 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
```

8.12 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.

8.12.1 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 that 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 will refuse 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 will negotiate 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 will cease 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.

8.12.1.1 Configuring RPC Authentication Levels

The default Cache Manager and File Server authentication settings are such that they will negotiate 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.

8.12.1.2 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 next reboot the machine.

To change the authentication levels temporarily, do the following:

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 valid arguments for *level* are defined in Section 5.12.1.1.

The following example command

- 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
```

8.12.1.3 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

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. Because both DCE LFS filesets and exported non-LFS file systems have entries in the Fileset Location Database (FLDB), the DFS Backup System can be used with both types of filesets.

This chapter introduces the DFS Backup System. It describes configuration issues related to the performance of backup and restore operations. Chapter 10 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 for information about backing up and restoring data.

9.1 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 System can be used to dump and restore data 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 that incompatible information may be lost when a fileset is dumped

and restored between file system types; 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. (Refer to your vendor's documentation to verify the level of support for dump and restore operations between different types of file systems.)

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 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.

9.1.1 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.

9.1.2 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 Section 9.3.3 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 Section 9.3.2 for a description of these characters and their interpretations.)

9.1.3 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.

9.1.4 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 Section 9.3.2 for more information.

9.1.5 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.

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.

9.2 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.

9.2.1 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 Part 2 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 256 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.

9.2.2 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.

9.2.2.1 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 Section 9.3.1 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.

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 *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 Section 9.3.2.for more information.)

9.2.2.2 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.

9.2.2.3 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 *dceshared*/**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. 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.
2. 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.
3. 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
```

9.2.2.4 Using the Interactive Interface

You can use the **bak** commands in regular command mode or in interactive mode. If you use interactive mode, not 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.

9.2.2.4.1 Entering Interactive Mode

Enter the **bak** command:

```
$ bak
```

9.2.2.4.2 Leaving Interactive Mode

Enter the **quit** command at the `bak>` prompt:

```
bak> quit
```

9.3 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
- Defining fileset families and fileset family entries
- Defining a dump hierarchy of dump levels
- Labeling tapes, if necessary

See Chapter 10 for information on inspecting the status of these prerequisites. If all of the prerequisites are met, turn to the appropriate section in Chapter 10 for information on using the Backup System.

9.3.1 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 Section 9.3.1.1 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 Section 9.3.1.2 are required for any machine to be configured as a Tape Coordinator machine.

9.3.1.1 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 Section 9.3.1.2; 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**/*hostname* directory, where *hostname* is the name

of the machine to be configured as a Tape Coordinator machine). 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/hostname/dfs-server
```

In the command, *hostname* is the name of the machine to be configured as a Tape Coordinator machine (for example, **client1**). 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/hostname/dfs-server \> -group subsys/dce/dfs-admin -org none \> \
-password acct_password -mypwd your_password
```

In the command, **hosts/***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/*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/*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/***hostname***/dfs-server \> **
   -add *{group subsys/dce/dfs-admin rcDnfmag}*
   dcecp> exit

Once you have completed these steps, perform all of the steps in Section 9.3.1.2.

9.3.1.2 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 Section 9.3.1.1 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 Section 9.3.1.1.

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 Section 9.2.2.3.

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 Section 9.1.5); 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 Section 9.2.2.4. 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.

9.3.2 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_**Vtape_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. Section 9.3.2.7 contains annotated sample scripts that illustrate typical routines to control automated backup equipment.

9.3.2.1 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. Section 9.3.2.4 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.

9.3.2.2 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**.

9.3.2.3 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.

9.3.2.4 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.

9.3.2.5 The NAME_CHECK Parameter

The **NAME_CHECK** parameter 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** 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.

9.3.2.6 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.

9.3.2.7 Sample User-Defined Configuration Files

The following sample **conf_**Vtape_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_**Vtape_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_**Vtape_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_**Vtape_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.

9.3.2.7.1 Sample conf_Vtape_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.

9.3.2.7.2 Sample conf_Vtape_device File for Jukeboxes

The following sample **conf_**Vtape_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_**Vtape_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** or **readlabel** operations are 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.

9.3.2.7.3 Sample conf_Vtape_device File for Dump to File

The following sample **conf_**Vtape_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** or **readlabel** operations are 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.

9.3.3 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.

9.3.3.1 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 9-1 summarizes some suggested groupings.

Table 9–1. 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	**".*" ".*" ".*"**
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.

9.3.3.2 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*

9.3.3.3 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.

9.3.3.4 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***...**

9.3.3.5 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.

9.3.4 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 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 256 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 for more information on the **bak lsdumps** command.)

9.3.4.1 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
```

9.3.4.2 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*...**]**

9.3.4.3 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*...

9.3.4.4 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*

9.3.5 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 Section 9.3.1 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 Section 9.3.2.5 for information about the **NAME_CHECK** parameter in the user-defined configuration file.

9.3.5.1 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 Section 9.2.2.1 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***]**

9.3.5.2 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 Section 9.2.2.1 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 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.

9.4 Adding and Removing Tape Coordinators

As mentioned in Section 9.3.1, 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.

Section 9.4.1 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 Section 9.3.1. You use the **bak addhost** command to add an entry for a Tape Coordinator to the Backup Database.

Section 9.4.2 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.

9.4.1 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 *dcelocal*/**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 Section 9.2.2.3.

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***]**

9.4.2 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 *dcelocal*/**var/dfs/backup/TapeConfig** file.
3. Using a text editor, remove the line that defines the Tape Coordinator from the *dcelocal*/**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 describes configuration of the Backup System in detail; make sure the Backup

System is properly configured according to the guidelines detailed in Chapter 9 before attempting to use it to perform any of the operations described in this chapter.

10.1 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, 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 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.

10.2 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.

10.2.1 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 Part 2 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.

10.2.2 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.

10.2.2.1 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 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 *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 only prompts for new tapes; it also displays some output as necessary for operations that it executes. This is the default.

10.2.2.2 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.

10.2.2.3 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.

10.2.2.3.1 Entering Interactive Mode

Enter the **bak** command to initiate interactive mode:

```
$ bak
```

10.2.2.3.2 Leaving Interactive Mode

Enter the **quit** command at the `bak>` prompt to terminate interactive mode:

```
bak>quit
```

10.2.2.4 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 for more information about these commands.)

10.2.2.5 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 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.
  - The `[abort requested]` message is displayed if the **kill** command was issued but the operation is not yet canceled.
  - The `[abort sent]` message is displayed if the operation is canceled but its execution is not yet stopped.
  - The `[operator wait]` message is displayed if the Tape Coordinator is waiting for the operator 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 Section 10.7.1 for information about the **bak jobs** command.)

10.3 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.

10.3.1 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.
```

10.3.2 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.*
```

10.3.3 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
```

10.3.4 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
```

10.3.5 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, TCID 0
Host /.../abc.com/hosts/bak1, TCID 1
Host /.../abc.com/hosts/bak2, TCID 3
Host /.../abc.com/hosts/bak3, TCID 8
Host /.../abc.com/hosts/bak3, TCID 7
Host /.../abc.com/hosts/bak3, TCID 6
```

10.3.6 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
```

10.3.7 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 Section 10.2.2.1 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 Section 10.6.3 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
.   .   .
```

10.4 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 at which the family is to be dumped. All entries in the specified fileset family are dumped according to the dump level that you specify. 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 data in the filesets that has changed since the filesets were dumped at the parent of the dump level that you specify 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 Section 10.4.3 for more information about the **bak deletedump** command.)

10.4.1 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.

10.4.2 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 Section 10.2.2.1 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 Section 10.2.2.3 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 for details.)

10.4.3 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 Section 10.2.2.3 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.

10.5 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.

You can use these commands to restore data to a file system that is different from the file system from which the data was dumped. For instance, data dumped from a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of non-LFS fileset. Similarly, data dumped from a non-LFS fileset can be restored to a DCE LFS fileset or to a different type of non-LFS fileset. (See your vendor's documentation to verify the level of support for dump and restore operations between different types of file systems.)

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.

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 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 Section 10.5.4 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.

10.5.1 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.

10.5.2 Restoring Individual Filesets

Use the **bak restoreft** command to restore one or more individual filesets. Table 10-1 summarizes the options available with the command. Unless indicated as Optional in the table, each option is required.

Table 10–1. 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**).
-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 Section 10.2.2.1 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 Section 10.2.2.3 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*/*yyhh*:*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.

10.5.3 Restoring an Aggregate with the bak restoredisk Command

Use the **bak restoredisk** command to restore all of the filesets on an aggregate. Table 10-2 summarizes the options available with the command. Unless indicated as Optional in the table, each option is required.

Table 10–2. 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 Section 10.2.2.1 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 Section 10.2.2.3 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.

10.5.4 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 Section 10.5.4.1.)
- 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 Section 10.5.4.2.)

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.

10.5.4.1 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 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 Section 10.2.2.1 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 Section 10.2.2.3 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.

10.5.4.2 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:

```
machineaggregatefileset [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 Section 10.2.2.1 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 Section 10.2.2.3 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.

10.6 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 *dcelocal***/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.

10.6.1 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 Section 10.2.2.1 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.

10.6.2 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 */dcelocal/* **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 */dcelocal* **/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 Section 10.2.2.1 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.

10.6.3 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 Section 10.2.2.1 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.

10.7 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.

10.7.1 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 Section 10.2.2.3 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.

  — The `[abort request]` message means the **bak kill** command was issued but the operation is not yet canceled.

  — The `[abort sent]` message means 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.

— The `[butc contact lost]` message means the **bak** program temporarily lost contact with the Tape Coordinator executing the operation.

— The `[drive wait]` message means the operation is waiting for the specified tape drive to become free.

— The `[operator wait]` message means the Tape Coordinator is waiting for the operator monitoring the command's execution to insert a tape in the drive.

10.7.2 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 Section 10.2.2.3 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.

11.1 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.

11.1.1 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 Section 11.1.2).

11.1.2 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.

11.1.3 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
```

11.1.4 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.

11.1.4.1 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 Section 11.1.3 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
```

11.1.4.2 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.

11.2 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.

11.2.1 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.

11.2.1.1 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.

11.2.1.2 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.

11.2.2 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.

11.2.2.1 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*).

11.2.2.2 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**.

11.2.3 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
```

11.2.4 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 sets -inactive
```

11.2.5 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)
```

11.2.6 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
```

11.2.7 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 *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 *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
```

11.2.8 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
```

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W