OS/390®

Distributed File Service DFS Administration Guide and Reference

SC28-1720-06

OS/390®

Distributed File Service DFS Administration Guide and Reference

SC28-1720-06

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix F, "Notices" on page 809.

Seventh Edition (December 1999)

| This edition applies to Version 2, Release 8 of OS/390 (product number 5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC28-1720-05,

The following statements are provided by the Open Software Foundation.

Contents

Figures

Tables

About This Book

The purpose of this book is to provide complete and detailed guidance and reference information for system and network administrators to use when they work with the IBM OS/390® Distributed File Service (OS/390 DFS) product.

OS/390 Distributed File Service includes a DFS function that is based on the OSF DCE Distributed File Service component. The OSF DCE Distributed File Service includes distributed file system capabilities based on the OSF DCE RPC protocols and security.

Who Should Use This Book

This book is intended for users and system and network administrators who understand the basic concepts of data communications. A knowledge of TCP/IP communications and the UNIX operating system can help you to use this guide more effectively. If you have little or no experience with Distributed Computing Environment (DCE), you should first read *Distributed Computing Environment: Understanding the Concepts*.

How This Book Is Organized

The information in this book is divided into two parts, each part divided into chapters.

Part 1, "OS/390 DFS Administration Guide" on page 1 discusses the guidance information. The chapters in that part begin with a short introduction and are followed by detailed information.

Part 2, "OS/390 DFS Reference" on page 325 discusses the reference information, organized into three chapters.

- Chapter 17, "OS/390 System Commands" on page 327 provides information regarding the MVS system commands, **MODIFY**, **START**, and **STOP**.
- Chapter 18, "Distributed File Service Files" on page 337 introduces the **DFS** files for OS/390 DFS.
- Chapter 19, "Distributed File Service Commands" on page 401 provides descriptions of **DFS** command syntax for OS/390 DFS.
- Chapter 20, "Distributed File Service Application Programming Interface" on page 771 introduces the DFS API, **w_pioctl**.

The chapters begin with a short introduction and are followed by information about the daemons, the control programs, the control program commands, which are ordered alphabetically, and finally the application programming interface.

In addition, there are the following appendices.

- Appendix A, "Environment Variables in DFS" on page 775.
- Appendix B, "Sharing Data Between OS/390 UNIX Applications, Commands, and DFS Clients" on page 795.
- Appendix C, "Examples of Working with Character Data from Other Platforms" on page 799.
- Appendix D, "DFS/NFS Secure Gateway" on page 805.
- | Appendix E, "Using Both SMB and DCE DFS" on page 807.

Note: This book lists some commands and processes that are not supported in OS/390 DFS. All such information is identified accordingly. Refer to “Important Information - Please Read Me!” on page 3 for more information on this.

Where to Find More Information

This book provides guidance and reference information for using OS/390 DFS. For information regarding configuring your system, see the *OS/390 Distributed File Service DFS Configuring and Getting Started* book. Information regarding commands and command syntax can be found in Part 2, “OS/390 DFS Reference” on page 325.

Where necessary, this book references information in other books using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of OS/390, see the *OS/390 Information Roadmap*, GC28-1727.

Unsupported Services and Commands

The following commands and services are not supported in OS/390 DFS:

- The **TapeConfig** configuration file.
- The **FMSLog** log file.
- The following **bos** commands are not supported in OS/390 DFS:
  - **bos getdates** command
  - **bos install** command
  - **bos prune** command
  - **bos uninstall** command.

  In addition, the **-newbinary** option is not available in OS/390 DFS for any **bos** commands.
- The following **bak** commands cannot be issued against an OS/390 **bakserver**. These commands may be issued from OS/390 against non-OS/390 **bakserver** processes:
  - **bak labeltape** command
  - **bak scantape** command
  - **bak readlabel** command.
- The **fms** command.
- The **dfsnfs** gateway function is provided, see Appendix D, “DFS/NFS Secure Gateway” on page 805. However, the following **dfsgw** commands cannot be issued from OS/390:
  - **dfsgw add** command
  - **dfsgw apropos** command
  - **dfsgw delete** command
  - **dfsgw help** command
  - **dfsgw list** command
  - **dfsgw query** command.
- The **dfstrace** command.

Commands and services not directly supported in OS/390 DFS may be run or requested from other non-OS/390 DFS systems. Exceptions are noted where applicable.

Conventions Used in This Book

This book uses the following typographic conventions:

Bold	**Bold** words or characters represent system elements that you must enter into the system literally, such as commands.
Italic	*Italicized* words or characters represent values for variables that you must supply.
`Example Font`	Examples and information displayed by the system are printed using an example font that is a `constant width typeface`.
[ ]	Optional items found in format and syntax descriptions are enclosed in brackets.
{ }	A list from which you must choose an item found in format and syntax descriptions are enclosed by braces.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on a keyboard.
...	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
\	A backslash is used as a continuation character when entering commands from the shell that exceed one line (255 characters). If the command exceeds one line, use the backslash character \ as the last non-blank character on the line to be continued, and continue the command on the next line.

This book uses the following keying convention:

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

Entering commands
When instructed to enter a command, type the command name and then press <**Return**>.

Online Books

All the books belonging to the IBM OS/390 Distributed File Service library are available as online publications. They are included in the *IBM OS/390 Collection*, SK2T-6700.

All the books in the Online Library are viewable, without change, on these IBM operating platforms: OS/390, VM, OS/2®, DOS, and AIX/6000®. The same book can be viewed on any of these platforms using the IBM BookManager® Library Reader™ for OS/2, Windows, and DOS, or any of the IBM BookManager READ licensed programs for OS/390, VM, OS/2, Windows, DOS, or AIX/6000.

The booklet, included with the Online Library, provides details on accessing the IBM OS/390 Distributed File Service online publications.

How to Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other OS/390 documentation:

- Visit the home page at:
  `http://www.ibm.com/s390/os390/`
- Fill out one of the forms at the back of this book and return it by mail, by fax, or by giving it to an IBM representative.

Summary of Changes

Summary of Changes
for SC28-1720-06
OS/390 Release 8

This book contains information previously presented in Release 7 of *OS/390 Distributed File Service Administration Guide and Reference*, SC28-1720-05.

New Information: The OS/390 Distributed File Service base element now supports both Distributed Computing Environment (DCE) DFS file protocols and the Server Message Block (SMB) file/print protocols. The SMB function is based on the X/Open SMB Version 2 specification and the IETF RFCs on Netbois over IP (RFC1001 and RFC1002). For more information specific to the SMB function refer to the *OS/390 Distributed File Service SMB Administration Guide and Reference*.

The following SMB files have been added:

- **smbidmap**
- **smbtab**.

The following SMB command has been added:

- **dfsshare**.

There are also additional environment variables that have been added that pertain specifically to SMB and an Appendix that explains how you use both DCE DFS and SMB together.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for SC28-1720-05
OS/390 Release 7

This book contains information previously presented in Release 6 of *OS/390 Distributed File Service Administration Guide and Reference*, SC28-1720-04.

Changed Information

- The following commands have been changed:
  - **dfsd**
  - **fxd**
- OS/390 supports DCE Local File System aggregate sizes that are greater than 4GB. The actual maximum size is based on the aggregate blocksize. Aggregates can contain 2G blocks. For a 4K blocksize, the aggregate can be 8TB (4K × 2G). For a 64K blocksize, the aggregate can be 128TB (64K × 2G).
- DFS server character data translation can be controlled on a file name suffix basis for all HFS data using the HFS attributes (**hfsattr**) file, and a devtab HFS entry option allows translation to be based on file contents.

New Information

- 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.
- 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.
- The following new commands have been added:
  - **cm getprotectlevels**
  - **cm setprotectlevels**
  - **fts setprotectlevels**.

Summary of Changes
for SC28-1720-04
OS/390 Release 6

This book contains information previously presented in Release 5 of *OS/390 Distributed File Service Administration Guide and Reference*, SC28-1720-03.

Changed Information

- As part of the name change of OpenEdition® to OS/390 UNIX System Services, occurrences of OpenEdition have been changed to OS/390 UNIX System Services or its abbreviated name, OS/390 UNIX. OpenEdition may continue to appear in message, panel text, and other code locations.
- The DFSKERN process of the OS/390 DFS File Server can be run in a separate address space. This can provide better recovery characteristics in DFSKERN failure situations.
- OS/390 DFS supports file sizes greater than 2 Gigabytes (or $2^{31}-1$ bytes). File sizes at the server are dependent on the Physical File System (PFS) being exported. File sizes at the client are dependent on the DFS client chunk size. The OS/390 DFS client supports up to 2 Gigabyte chunks. (Chunks are generally 64K for disk caching or 8K for memory caching. This would result in maximum file sizes of $(2^{47})-1$ bytes and $(2^{44})-1$ bytes, respectively.)
- Multiple DFS filesets on a DFS backup tape are contained in a single tape data set. Previously, each DFS fileset was a separate tape data set.
- Cache Manager translation can now be controlled on a file name suffix basis using the **cm attributes** (**cmattr**) file, and a Cache Manager **-translation** option allows translation to be based on file contents.

Summary of Changes
for SC28-1720-03
OS/390 Release 5

This book contains information previously presented in Release 3 of *OS/390 Distributed File Service Administration Guide and Reference*, SC28-1720-02.

The following summarizes the changes to that information.

New Information: The following new commands have been added:

- **cm getpreferences**
- **cm setpreferences**.

The following file has been added:

- **CMLog**

The following new Application Programming Interface (API) has been added:

- **w_pioctl** - reads a DFS mountpoint

Changed Information: The following commands have been changed:

- **dfsd**
- **fts release**.

Information on the following has changed:

- The maximum DCE Local File System aggregate size is now 4GB.
- An authorized user can now release an RFS file by using the **touch** command.
- RFS attributes can now be specified on a file basis (by using the Attributes file).
- The **dfstrace** command suite has been removed.

Summary of Changes for SC28-1720-02 OS/390 Release 3

This book contains information previously presented in Release 2 of *OS/390 OpenEdition DCE DFS Administration Guide and Reference*, SC28-1720-01.

The following summarizes the changes to that information.

New Information: Information on the following has been added and is now supported in Release 3 of OS/390:

- The DFS Server exports OS/390 record data.

The following new chapter has been added:

- Chapter 12, "Using OS/390 Data Sets from DFS Clients" on page 235. This chapter explains what you need to know when you use OS/390 record data sets from a DFS client workstation.

The following new DFS file has been added:

- "Attributes File (rfstab)" on page 341.

Changed Information: Updates for record data have been made to Chapter 10, "Making Filesets and Aggregates Available" on page 147.

Updates have also been made to the following files:

- "devtab" on page 368
- "dfstab" on page 372.

Part 1. OS/390 DFS Administration Guide

This part of the book discusses the guidance information in the following chapters:

Each chapter begins with a short description and is followed by detailed information.

Chapter 1. An Overview of DFS

This chapter introduces basic concepts of the 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 2, "OS/390 DFS Reference" on page 325.

Note: This chapter is a general understanding of DFS and is not necessarily how it specifically works on OS/390.

Important Information - Please Read Me!

The purpose of this book is to provide complete and detailed guide information for DFS. Not all the DFS features, commands, daemons, or concepts described in this book are supported in OS/390 DFS on the OS/390 operating system. These parts of DFS that are not supported are indicated by notes in label boxes like this:

Important Note to Users

A labeled box, such as this, points out differences between DFS provided by OSF as described in this book, and DFS as supported for OS/390 DFS.

In some cases, where there are exceptions throughout the chapter, to preserve readability of this book, exceptions are noted at the beginning of the chapter in a labeled box. When using this book **it is important to check the beginning of the chapter** you are referring to and be aware of exceptions noted there. They are not noted everywhere they appear in the book.

UNIX Mode Bits and UNIX Commands

References to UNIX mode bits in this book also apply to the OS/390 UNIX System Services (OS/390 UNIX) implementation of POSIX file mode bits.

In general, examples showing UNIX commands apply to the OS/390 UNIX implementation of POSIX commands.

Root User ID References

In general, references to the root user ID in OS/390 UNIX are equivalent on OS/390 to a TSO ID with a **UID = 0**.

DCE Local File System

This book makes references to the DCE Local File System which refers to the file system provided by OSF. The following commands and server apply only to DCE Local File System:

- **newaggr** command
- **growaggr** command
- **salvage** command
- the **repserver** server.

References to a non-Local File System refer to the Hierarchical File System (HFS) and the Record File System (RFS).

HFS is the Hierarchical File System for byte data. RFS is a virtual file system that presents OS/390 data sets as a (pseudo-)hierarchical file system of byte data to DFS clients.

Your Local File System

References in this book to your local file system refer to HFS.

Comparison between Local File System and Other File Systems

Comparisons of Local File System to other file systems in this book generally apply to workstation file systems and do not necessarily apply to OS/390 HFS.

About the Examples in This Book

In IBM OS/390 Distributed File Service the administration and user commands can be submitted as a batch job, run from TSO or the OS/390 shell. For simplicity, the examples in this book are shown in shell mode. That is, commands are *entered* from the shell. The shell prompt is shown in examples as the dollar sign ($). In instances where *root* authority is needed (in OS/390 DFS, **root** refers to a user with a **UID = 0**), a number sign (#) is used for the command prompt. There is a slight difference in some command names used to run these facilities when running from TSO (or batch) and from the OS/390 shell.

While OS/390 DFS commands can be entered in either uppercase or lowercase in TSO (or batch), these commands can only be entered in lowercase in the OS/390 shell.

In batch, these names correspond to the data set members names that are shipped with the OS/390 DFS product for these facilities. For specific information on OS/390 DFS commands, see Part 2, "OS/390 DFS Reference" on page 325.

Commands that Cannot Fit on One Line

When entering commands from the shell, the command may exceed one line (255 characters). If the command exceeds one line, use the backslash character (\) at the end of the line to be continued, and continue the command on the next line.

Important Note to Users

When you enter a command from this book that uses the backslash character (\) make sure you immediately press the Enter key and then continue with the rest of the command. In most cases, the backslash has been used for ease of readability.

DFS Features

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

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

DFS Server Machines

DFS server machines run processes that provide services such as making data available and monitoring and controlling other processes. They are categorized by the processes they run (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 4, “DFS Configuration Issues” on page 43 for more information about the roles of DFS server machines.)

DFS Client Machines

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

Client machines use a modified kernel that maintains contact with the File Exporter and server processes running on server machines. This collection of kernel modifications on a client machine is known as the Cache Manager. The main duty of the Cache Manager is to translate file requests made by application programs on a client machine into Remote Procedure Calls (RPCs) to File Exporter processes on File Server machines.

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

DFS Data Access Management

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

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

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

DFS Administrative Domains

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

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

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

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

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

DFS Administrative Lists and Groups

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

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

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

Similarly, you can create additional groups for other administrative tasks, such as managing processes or installing new system binaries, and include the same or different users in these groups. Users have only the privileges associated with the administrative lists in which they are included. Unless users are also members of other administrative lists in a domain or in the cell to which a domain belongs, their membership in an administrative list on a machine grants them no additional privileges beyond the scope of that administrative list. You can limit a group's administrative duties by placing it on only certain administrative lists in a domain.

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

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

DCE Local File System

The DCE Local File System is a high-performance, log-based file system. DCE Local File System supports the use of aggregates. A DCE Local File System 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 Local File System 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 Local File System 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 Local File System log is damaged. To detect and repair inconsistencies in the file system that are not repairable by the log mechanism, 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.

Important Note to Users

The **fsck** program is not available in OS/390 UNIX.

DCE Local File System aggregates also support the use of filesets. A DCE Local File System fileset is a hierarchical grouping of files managed as a single unit. DCE Local File System filesets can vary in size but are almost always smaller than a disk partition. With DCE Local File System, multiple filesets can be stored on a single aggregate, providing flexible disk usage. A non-DCE Local File System (for example, an OS/390 HFS data set) 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-Local File System aggregate* and *non-Local File System fileset* are used to refer to exported non-Local File System partitions and the file systems they contain.)

The unique metadata structure of DCE Local File System aggregates also supports additional fileset operations not found on standard, non-Local File System partitions. With DCE Local File System, the potentially small size of filesets allows them to be easily managed for maximum system efficiency. Also, each DCE Local File System aggregate can store multiple DCE Local File System filesets. A system administrator can move filesets from one DCE Local File System 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 Local File System fileset corresponds logically to a directory tree in the file system. Each fileset maintains, on a single DCE Local File System aggregate, all of the data that comprises 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 Local File System 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 Local File System 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 Local File System also supports the use of DCE ACLs to set permissions on directories and files in DCE Local File System filesets. DCE ACLs extend the standard UNIX permissions, which are set with UNIX mode bits, to offer more precise definitions of access permissions for directories and files. ACLs and administrative lists restrict access to DFS management operations in a cell or domain to specifically authorized users.

DFS Replication

DCE Local File System allows you to replicate (copy) DCE Local File System filesets. When you replicate a DCE Local File System 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 Local File System fileset. Replication is supported only for DCE Local File System filesets, not for non-Local File System filesets (file systems on non-Local File System partitions).

Two types of replication are available with DCE Local File System: 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.

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 Local File System filesets and non-Local File System filesets are supported.

Important Note to Users

OS/390 DFS does not support backing up HFS. See the *OS/390 UNIX System Services User's Guide*, SC28-1891, for details on how to back up HFS. OS/390 DFS does not support backing up OS/390 data sets (RFS). See the *DFSMS/MVS DFSMShsm Storage Administration Guide*, SH21-1076, for details on how to backup OS/390 data sets.

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.

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 Local File System fileset and mount it as a subdirectory of the user's home directory, naming it something appropriate such as **.OldFiles** or **.BackUp**. The user can then, without assistance, restore to a read/write fileset any files deleted or changed since the backup fileset was made. Users cannot change the data in their backup filesets, but they can copy the data to a regular directory in a working, read/write fileset and use it there.

DFS Database Distribution

DFS houses fileset and Backup System information in two administrative databases:

- The **Fileset Location Database (FLDB)** stores information about the locations of filesets, and
- The **Backup Database** records information about backups and tapes.

To maintain file system reliability and availability, the two databases are replicated on multiple server machines. If any of the machines housing a database becomes unavailable, the database is still available from other machines.

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

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

The DFS scout Program

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

Important Note to Users

Scout on the OS/390 operating system does not use a graphical display or highlighting.

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.
- The number of kilobytes available on aggregates on the File Server machine on which the File Exporter is running.

Accessing DFS from NFS

The DFS/NFS Secure Gateway provides authenticated access to DFS through 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 D, “DFS/NFS Secure Gateway” on page 805 for information about configuring and using the DFS/NFS Secure Gateway.)

Advantages of DFS

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

Faster Restarts and Better Reliability

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

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 manage this complex process:

- To ensure that users are always working with the most-recent copy of a file.
- To track who is currently working with the file.
- To identify operations the client can perform on the data.
- To ensure that the File Exporter 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 Local File System filesets on multiple File Server machines. When you replicate a fileset, you place an identical copy of the fileset on a different File Server machine. The unavailability of a single server that houses the fileset generally does not interrupt work involving that fileset because the fileset is still available from other machines.

Better Recovery from Failure

As detailed previously, recovering from an abnormal system shutdown is easier because DCE Local File System 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 Local File System 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 Local File System aggregate on which the file system is stored; it detects and repairs inconsistencies in the file system's metadata to return the file system to a consistent state.

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

Improved File Availability, Access Time, and Network Efficiency

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

- Replication increases file availability by allowing DCE Local File System filesets to be replicated on multiple server machines; this minimizes the effects of machine outages. If one machine housing a read-only copy of a DCE Local File System fileset is unavailable, other replicas of the fileset are usually still available from other machines.
- Caching data locally decreases access time to the data. The cache is an area of a client machine's local disk or memory dedicated to temporary data storage. Once data is cached, subsequent access to it is fast because the client machine does not need to send a request for it across the network. Thus, caching also minimizes network traffic. As noted previously, DFS ensures that each client housing cached read/write data always has access to the most-recent version of the data; in the case of cached read-only data, DFS ensures that each client has access to the most-recent version of the data within a configurable period of time.
- Multihomed File Servers both help administrators make efficient use of their networks and increase file availability. Network efficiency is improved by allowing administrators to create connections between file servers and the subnetworks or networks wherein most of the DFS clients reside. Multiple network connections per File Server also increases file availability in that a fault in one section of the network is less likely to make a File Server unavailable.

Efficient Load Balancing and File Location Transparency

Load balancing of data is more efficient in DFS than in standard nondistributed file systems. One reason is the use of replication, which allows DCE Local File System filesets to be replicated on multiple machines. Requests for files from frequently used DCE Local File System filesets are then spread across different machines, preventing any one machine from becoming overburdened with data requests.

Fileset characteristics in DCE Local File System also improve load balancing. DCE Local File System filesets are typically smaller than standard UNIX and other non-Local File System filesets; DCE Local File System aggregates can accommodate multiple DCE Local File System filesets for flexible disk usage; and DCE Local File System 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 Local File System or non-Local File System). Therefore, the user never needs to know the machine or aggregate that actually houses the fileset.

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

Extended Permissions

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

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

Increased Interoperability and Scalability

Data from non-Local File Systems can be used with DFS. You can export a non-Local File System disk partition to the DCE namespace for use as an aggregate in DCE. While an exported partition can be accessed in the namespace, it still holds only the single file system it contained at the time it was exported. Additionally, a non-Local File System aggregate may not support features such as logged information about metadata, DCE ACLs, and fileset replication, which are available with DCE Local File System.

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

Increased Security and Administrative Flexibility

DFS supports enhanced administration and security by making DCE ACLs available with objects in DCE Local File System 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 also enable or disable the honoring of **setuid** and **setgid** programs on a per-fileset and per-Cache Manager basis. Thus, you can direct a specific Cache Manager to enable **setuid** and **setgid** programs located in a specific fileset (such as one that stores system binary files).

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

Consistency of Configuration and Binary Files

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

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

The DFS Update Server process distributes common files from System Control and Binary Distribution machines. Server machines that rely on System Control and Binary Distribution machines for configuration and binary files run the client portion of the Update Server (the **upclient** process). The System Control and Binary Distribution machines run the server portion of the Update Server (the **upserver** process).

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

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

System Monitoring

DFS provides two 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.

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 *Understanding OSF DCE 1.1 for AIX and OS/2* book, SC24-4616, and understands the following DCE components described in the *OS/390 DCE Administration Guide*:

- DCE Security Service, especially the keytab file and ACLs.
- DCE Directory Service, especially details about the namespace.
- DCE Distributed Time Service, especially client and server machine synchronization.
- DCE Remote Procedure Call facility, especially client and server machine communications.

Note: Many of these services also interact with each other.

DCE Security Service

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

- **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.
- **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.
- **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:

- **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.
- **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.

  OS/390 DFS also provides an **AUTOLOGIN** function, see “Controlling the DFS Client Usage of the OS/390 DCE Single Sign-on Function” on page 273.

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 Directory Service has three main components:

- **Cell Directory Service (CDS)** manages names within a cell. Each resource has a CDS entry that is unique within its local cell.
- **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.
- **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 through either GDS or the Domain Name System (DNS), a widely used global naming environment.

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

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

The second example shows a similar entry in GDS format:

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

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

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

DFS Filespace: The default name for the root of a DCE cell's DFS filespace is **fs**, which is an entry in the cell's namespace. The **fs** entry, referred to as a junction, serves as a boundary between the CDS namespace and the DFS filespace. The contents of the **fs** junction provide the information necessary to access files and directories in the filespace. (Note that the terms DCE namespace and DFS namespace can be used interchangeably.)

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

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

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

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

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

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

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

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

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

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

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 UNIX; in this case, the **:** is used to separate different pathnames, so including a prefix in the definition of a **PATH** environment variable violates the reserved nature of the **:** for that variable.

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

DCE Distributed Time Service

The DCE Distributed Time Service (DTS) provides precise synchronization for system clocks in a network. In DFS, clock synchronization is important for communications between client machines using the Cache Manager and server machines running the File Exporter and other server processes. Clients and servers must refer to a common time standard for communications to remain constant and for data to remain available.

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

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

DCE Remote Procedure Call

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

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

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

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

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

System Administration: A Task Overview

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

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

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

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

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

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

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

DFS also includes a number of miscellaneous, nonsuite commands; for example, the **scout** command is a miscellaneous command that 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.

Important Note to Users

In OS/390 DFS, the **dfsgw** commands are not available. See Appendix D, “DFS/NFS Secure Gateway” on page 805 for more information on the DFS/NFS Secure Gateway.

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

Refer to Part 2, “OS/390 DFS Reference” on page 325 for detailed discussions of the various DFS commands. Refer to the *OS/390 DCE 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.

Fileset Management Commands

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

Fileset (fts) Commands:

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

- Create a read/write fileset with the **fts create** command; mount the fileset in the file tree with the **fts crmount** command.
- Examine a mount point with the **fts lsmount** command; delete a mount point with the **fts delmount** command.
- Create a backup version of a single fileset with the **fts clone** command; create backup versions of many filesets at once with the **fts clonesys** command.
- Prepare to replicate a fileset by assigning replication parameters with the **fts setrepinfo** command.
- Define replication sites with the **fts addsite** command; remove replication sites and read-only replicas at the sites with the **fts rmsite** command.
- Create read-only replicas of a fileset with the **fts release** and **fts update** commands.
- Check the status of the Replication Server on a File Server machine with the **fts statrepserver** command; check the status of each replica of a fileset with the **fts lsreplicas** command.
- Set a fileset's quota with the **fts setquota** command; list the quota with the **fts lsquota** command.
- List fileset header information with the **fts lsheader** command.

- List FLDB information with the **fts lsfldb** command.
- Examine fileset header information and FLDB information with the **fts lsft** command.
- Move a fileset with the **fts move** command.
- Remove a fileset with the **fts delete** command.
- Dump a fileset to a byte stream format with the **fts dump** command; restore a fileset to the file system with the **fts restore** command.
- Set advisory RPC authentication bounds on a per-fileset basis with the **fts setprotectlevels** command.

Backup (bak) Commands: You can use **bak** commands to perform the following types of tasks:

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

Important Note to Users

When OS/390 DFS uses MVS services to control and dynamically allocate tape drives, the **bak** commands: **bak labeltape**, **bak readlabel**, and **bak scantape** are not necessary in OS/390 DFS and are not available.

See “bak” on page 408 for more information on the **bak** commands.

System Management and Configuration Commands

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

Cache Manager (cm) Commands: You can use **cm** commands to perform the following types of tasks:

- List the current cache size and type with the **cm getcachesize** command.
- Set the size of the cache with the **cm setcachesize** command.
- Force the Cache Manager to discard cached data and information about the data with the **cm flush**, **cm flushfileset**, and **cm checkfilesets** commands.
- Determine the Cache Manager preferences for File Server machines from which to access read-only filesets by using the **cm getpreferences** command.

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

Basic OverSeer (bos) Commands: You can use **bos** commands to perform the following types of 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.

Important Note to Users

The following **bos** commands are not available in OS/390 DFS:

- **bos getdates** command
- **bos install** command
- **bos prune** command
- **bos uninstall** command.

See “bos” on page 475 for more information on the **bos** commands.

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

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

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

Security Commands and Tools

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

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

You can also use the following **dcecp** commands 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 options (specified in the DFSCM **_IOE_CM_PARMS** environment variable, refer to "Cache Manager Initialization Parameters" on page 273) 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 options (specified in the **ioepdcf** file, refer to "ioepdcf" on page 382).

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.

Important Note to Users

In OS/390 DFS, the **dfs_login**, **dfs_logout**, and **dfsgw** commands are not available. See Appendix D, “DFS/NFS Secure Gateway” on page 805 for more information on the DFS/NFS Secure Gateway.

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:

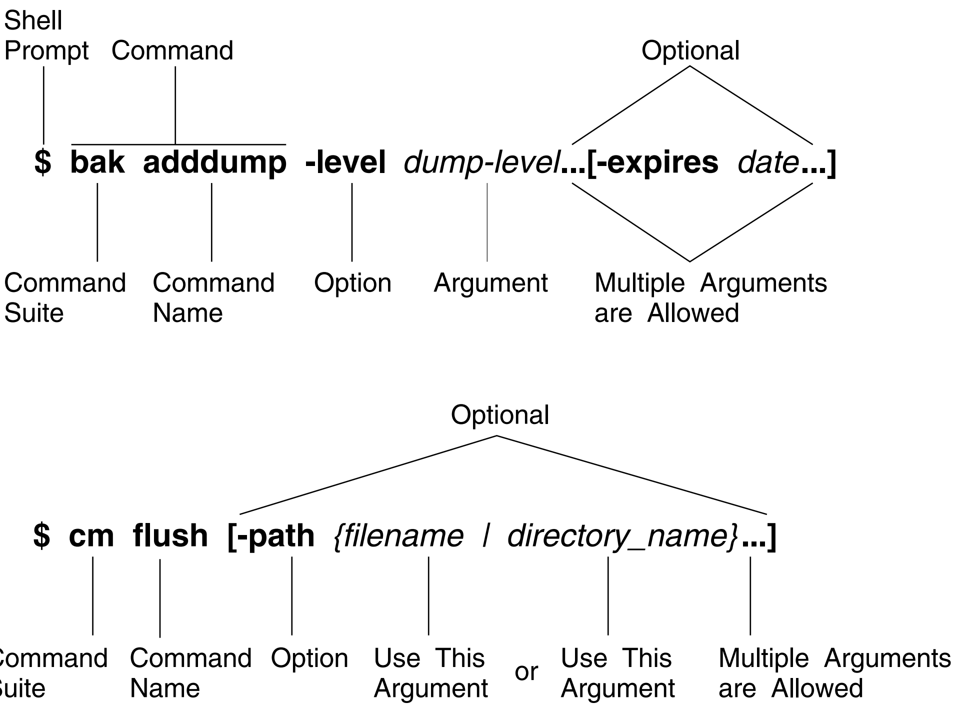

Shell Prompt
Command
Optional
$ bak adddump -level dump-level...[-expires date...]
Command Suite
Command Name
Option
Argument
Multiple Arguments are Allowed
Optional
$ cm flush [-path {filename | directory_name}...]
Command Suite
Command Name
Option
Use This Argument
or
Use This Argument
Multiple Arguments are Allowed

Figure 1. Elements of a DFS Command

The following list summarizes the elements of a DFS command:

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

Options Command options always appear in bold font in the text, are always preceded by a **-** (dash), and are often followed by arguments. In the first example, **-level** and **-expires** are options, and *dump_level* and *date* are their arguments; in the second example, **-path** is the only option.

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

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

Optional Information

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

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

Command Shortcuts

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

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax. The syntax for each command is presented with its description in Part 2, "OS/390 DFS Reference" on page 325.
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.
- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

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

If 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
  ```

Receiving Help

There are several different ways to receive help about DFS commands. The following list summarizes the syntax for the different help options available in OS/390 DFS:

- 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 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 DFS commands that are not part of a suite.

Chapter 2. DFS on OS/390

DFS can run on several different operating systems, however, not all components of DFS as designed by OSF work on all operating systems. This chapter focuses on differences between the OS/390 UNIX implementation of DFS and the OSF implementation of DFS.

Commands Ported to OS/390 DFS

Not all commands or command functions supported in the OSF version of DFS have been ported to OS/390 DFS. The following table lists the OSF command or command suite with a brief description and indicates whether the command or its equivalent function is available in OS/390 UNIX. Note that in OS/390 DFS some of the commands within a command suite are not available. See "Unsupported Services and Commands" on page xxiv for more information.

Table 1. Availability of DFS Commands

Command	Available in OS/390 DFS?	Purpose
bak	yes	Used by system administrators to work with the DFS Backup System.
bos	yes	Used by system administrators to contact the Basic OverSeer (BOS) server.
cm	yes	Used by system administrators to set current Cache Manager configuration parameters. In addition, this command is used by general users to prompt various Cache Manager functions and request various information.
dfsexport	yes	Used by system administrators to prompt the File Exporter to export or unexport file aggregates.
dfsgw	no	Used to manipulate entries in the local authentication table on a Gateway Server machine. See Appendix D, "DFS/NFS Secure Gateway" on page 805 for more information.
dfstrace	no	Used by system administrators to diagnose problems within the DFS kernel or within the server processes that interface with the **dfstrace** command suite.
fms	no	Used with the Backup System to determine the tape size and EOF mark size for the tape drive indicated with **-device**.
fts	yes	Used to instruct the Fileset Server to create and delete filesets, as well as to move, replicate, and backup filesets.
growaggr	yes	Used to increase the size of an existing DCE Local File System aggregate.
newaggr	yes	Used to initialize a partition on the local disk of a machine for use as an aggregate with DCE Local File System.
salvage	yes	Used to recover, verify, or salvage the structure of a DCE Local File System aggregate.
scout	yes	Displays statistics gathered from the File Exporter running in the kernel on each File Server machine specified with **-server**.
udebug	yes	Used to display Ubik status information on the specified server in the specified RPC server group.

Daemons Ported to OS/390 DFS

Not all daemons supported in the OSF version of DFS have been ported to OS/390 DFS. The following table lists the DFS daemons with a brief description and indicates whether the daemon, or its equivalent function, is available in OS/390 UNIX.

Table 2. Availability of DFS Daemons in OS/390 DFS

Daemon	Available in OS/390 DFS?	Role
bakserver	yes	Communicates with the backup database to perform dump and restore operations.
bosserver	yes	Monitors other DFS server processes, such as the **flserver** and **ftserver** processes running on the machine, and restarts failed processes automatically.
butc	yes	Provides coordination of system tape machines for backup and restores.
dfsbind	no[1]	Provides user-space service to the cache manager on a DFS client machine or the File Exporter on a DFS File Server machine.
dfsd	yes	Initializes the DFS Cache Manager (**DFSCM**) on an OS/390 system according to information supplied by the **dfsd** command options supplied as **DFSCM** initialization parameters. The **dfsd** daemon functions are provided by the **ioecmini**, **ioedfsd**, and **ioelogin** processes running in the **DFSCM** address space.
dfsgwd	no[1]	Initializes the Gateway Server process. See Appendix D, "DFS/NFS Secure Gateway" on page 805 for more information.
flserver	yes	Maintains the Fileset Location Database (FLDB) that tracks the location of all DCE Local File System and non-LFS filesets.
ftserver	yes	Handles fileset administration operations such as creating, deleting, moving and replicating filesets.
fxd	yes	Uses information passed to it such as local cell name, local Fileset Database machine information and the identity of the group that can administer the File Exporter, to communicate with other processes such as **flserver** to ensure that only privileged users administer data in filesets exported from this machine.
repserver	yes	Tracks currency of replicas and updates the versions of data being used at each replication site.
upclient	yes	Initializes the client portion of the update server.
upserver	yes	Distributes files such as common configuration files, and administrative lists from System Control machines to other server machines in a domain.

OS/390 DFS Considerations

This section discusses the OS/390 DFS considerations.

[1] Although this daemon is not available, the function is provided in OS/390 DFS.

Working Directory

When you display your current working directory from the shell, (for example with **pwd**), it is recommended that you set the shell's logical flag (with **set -o logical**). This specifies that **cd**, **pwd**, and the **PWD** variable use logical pathnames in directories with symbolic links. (See the *OS/390 UNIX System Services Command Reference*, SC28-1892, for more information on the **set** command.) In some cases, the physical working directory pathname cannot be properly constructed. When a user does not have read or search permission on some component of the working directory's pathname, the physical working directory pathname may not be able to be properly constructed.

DCE Credentials and the DFS Client

In order to do authenticated access to data through DFS, the DFS Client must associate the requestor with its DCE credentials[2]. This association is stored internally with the user. In order to create this association, the DFS client must find the requestor's DCE credentials. A user's DCE credentials are created when a user issues a **dce_login**. They are stored in the user's home directory[3].

Note: DCE credentials do not necessarily disappear when a user logs off TSO. DCE credentials are stored in a file in HFS and can be found again when the user logs back on to TSO. However, they do expire after the default ticket lifetime has elapsed.

When the first file request is made by a user, the DFS client finds the requestor's DCE credentials (by looking in the user's home directory) and stores an association with the user[4]. Thereafter, when subsequent file requests are made by the user, the DFS client uses the association stored with the user to determine the requestor's credentials.

DCE Login Processing: DCE login processing provides several mechanisms to control how DCE credentials are found. The default is that they are found in the user's home directory. The mechanisms include the following:

- **EUVSKRB5 DD** statement
- **_EUV_SEC_KRB5CCNAME_FILE** environment variable
- **_EUV_HOME** environment variable.

Refer to the *OS/390 DCE Administration Guide* for more information on these mechanisms.

None of these mechanisms are effective for the DFS client because the DFS client does not have access to the requestor's environment. The only way that DCE credentials can be found by the DFS client is by the user's home directory (the one specified in the user's RACF® profile in the OMVS segment).

Note: Although RACF is mentioned, any OS/390 external security manager (ESM) that has equivalent support can be used instead of RACF.

There are three methods of obtaining DCE credentials, they include the following:

- **dce_login** command
- DCE single sign-on
- DCE single sign-on from the DFS client.

2 DCE credentials contain a user's DCE UUID (Universal Unique IDentifier).

3 Actually, there is a file in the user's home directory called **krb5ccname** that *points to* (contains the name of) the actual credentials cache file (ccf). All ccfs are kept in the same directory **(/opt/dcelocal/var/security/creds)** and each ccf is owned by the user that issued the **dce_login**.

4 For historical reasons, this association is called a process authentication group (PAG).

dce_login Command: The **dce_login** command can be used to explicitly obtain DCE credentials for a particular DCE principal.

DCE Single Sign-On: In conjunction with RACF interoperability, DCE provides users the ability to effectively sign on (log in) to both OS/390 and DCE in one operation. OS/390 DCE single sign-on allows an OS/390 user who has already been authenticated to RACF to be logged in to DCE. DCE does this automatically when a DCE application is started in an address space and the user is not already logged in to DCE. OS/390 DCE provides cross-linking utilities to assist the administration of DCE single sign-on. (See the *OS/390 DCE Administration Guide* for more information on DCE single sign-on.)

If DCE single sign-on occurs before the first file request that involves the DFS client, then these are the credentials that are associated with the user.

DCE Single Sign-On from the DFS client: In certain cases, a user can obtain DCE credentials during DFS client processing without having done a **dce_login** command and without having done a DCE single sign-on (by invoking a DCE application).

When a user makes a file request that involves the DFS client (for example, **cd /...**), and the user has no credentials, or they have expired, a DCE single sign-on is attempted from the DFS client under the following conditions:

- The DFS client has not already stored an association with the user;
- The OS/390 user has a DCE segment in the RACF user profile that has a valid DCE principal and password and has specified AUTOLOGIN(YES); and
- The DFS client does not have the **_EUV_AUTOLOG=NO** environment variable specified (in the **/opt/dfslocal/home/dfscm/envar** file).

  Note: The **_EUV_AUTOLOG** environment variable setting in the *user's* environment variable has no effect on the DFS client processing because the DFS client does not have access to the user's environment variables.

The DCE single sign-on invoked by the DFS client also causes the DFS client to store an association with the user.

Recommended Login Methods: The following are recommended methods for obtaining DCE credentials:

- The DFS client DCE single sign-on method.
  1. The administrator has ensured that the OS/390 user has a DCE segment in the RACF user profile that is enabled for DCE single sign-on (see “RACF Segments for DFSCM DCE Single Sign-On Processing” on page 31 for the RACF commands required) and that the DFS client (**DFSCM**) is enabled for DCE single sign-on.
  2. The user has stored their DCE password into RACF using the **storepw** command.
  3. The OS/390 user logs on to TSO and enters the OMVS environment, and
  4. The user changes directories to a directory in the DCE namespace (for example, **cd /...**).
- The explicit login method.

  Use this method if your installation has not enabled your OS/390 ID for DCE single sign-on.
  1. The OS/390 user logs on to TSO and enters the OMVS environment.
  2. The user issues **klist** to determine if the user has unexpired DCE credentials (the expiration date and time are listed as 'Identity Info Expires: yy/mm/dd:hh:mm:ss').

     If they are unexpired, the user issues **kinit** and responds to the password prompt, or

If they are expired or non-existent, the user issues **dce_login** specifying the user's principal and responds to the password prompt.

| **Note: When you dce_login as a different DCE principal from the same OS/390 ID, the**
| **previous DCE credentials are destroyed. If you want to go back to the previous DCE**
| **principal, you must dce_login as that DCE principal again.**

3. The user changes directories to a directory in the DCE namespace (for example, **cd /...**).

Each of these methods assumes that the user has been enrolled into TSO, has an OMVS segment in the user's profile (with a UID and a home directory), and has been enrolled into the DCE registry (has a DCE principal and password).

RACF Segments for DFSCM DCE Single Sign-On Processing

Note: This section assumes that you are using the IBM RACF support. For further information on using the RACF commands that are referenced in the following instructions, refer to the *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919. If you are using an equivalent security support product, refer to the appropriate documentation to perform the equivalent functions.

To allow **DFSCM** to perform DCE single sign-on for an OS/390 user, the OS/390 user must be set up to exploit the RACF support for OS/390 DCE single sign-on to DCE feature. Refer to the *OS/390 DCE Administration Guide* for more information on single sign-on support.

Setting up an OS/390 user for DCE single sign-on requires defining a RACF segment and storing the DCE password for the OS/390 user. If set up properly, the OS/390 user is automatically logged into DCE when the DFS client processes the first file request for the user unless the DCE environment variable **_EUV_AUTOLOG=NO** is set in **/opt/dfslocal/home/dfscm/envar**.

An outline of the steps required to set up the OS/390 system to have **DFSCM** exploit DCE single sign-on processing are:

- Activate class KEYSMSTR

  ```
  SETROPTS CLASSACT(KEYSMSTR)
  ```

- Activate class DCEUUIDS

  ```
  SETROPTS CLASSACT(DCEUUIDS)
  ```

- Define entry DCE.PASSWORD.KEY in class KEYSMSTR being sure to supply a 16 position KEYMASKED value

  ```
  RDEFINE KEYSMSTR DCE.PASSWORD.KEY +
   SSIGNON(KEYMASKED(nnnnnnnnnnnnnnnn))

  Example:
  RDEFINE KEYSSTR DCE.PASSWORD.KEY SSIGNON(KEYMASKED(0034639986ACCFDE))
  ```

- Define DCE RACF segment for each **os390_user_id** that requires the DCE single sign-on capability.
  - Determine the **dce_user_principal_name** for the **os390_user_id**

    You can obtain the **dce_user_principal_uuid** for the **dce_user_principal_name** using the command:

    ```
    dcecp -c principal show dce_user_principal_name
    (or use the dcecp command dialog method)
    ```

  - Determine your DCE cell name (**dce_cell_name**).

    Obtain the **dce_cell_uuid** using the command:

```
klist
```

Define the DCE RACF segment using the command:

```
ALTUSER os390_user_id DCE(DCENAME('<dce_user_principal_name>') +
                          UUID(<dce_user_principal_uuid>) +
                          HOMECELL('/.../<dce_cell_name>') +
                          HOMEUUID(<dce_cell_uuid>) +
                          AUTOLOGIN(YES))
```

A complete example of this command is:

```
ALTUSER g1dfst2 DCE(DCENAME('g1dfst2') +
                    UUID(0000007f-6df2-21cf-9f00-10005ab1b41d) +
                    HOMECELL('/.../dcecell16.endicott.ibm.com') +
                    HOMEUUID(c65760b6-6c56-11cf-8c8a-10005ab1b41d) +
                    AUTOLOGIN(YES))
```

- You can display the DCE RACF segment for a user by using the command:

  ```
  LISTUSER os390_user_id NORACF DCE
  ```

  A complete example of this command is:

  ```
  LISTUSER g1dfst2 NORACF DCE
  ```

- To activate DCE single sign-on processing for an OS/390 user, each user must issue the following command from TSO:

  ```
  storepw dce_login_password dce_login_password
  ```

  Notes:

  1. The dce_login_password is specified twice in the **storepw** command. Refer to the *OS/390 DCE Command Reference*, for a discussion of the **storepw** command.
  2. The **storepw** command may return a message indicating that a DCE single sign-on login error occurred or that DCE could not acquire the correct DCE password, but a subsequent message may indicate that the **storepw** command was successful.
  3. If the **storepw** command is not successful, insure that the previous **SETROPTS** and **RDEFINE** commands completed successfully.

Fileset Move

If a DCE Local File System fileset is being moved from one OS/390 system to another OS/390 system, it is recommended that the **fts move** command be issued from an OS/390 system.

Maximum File Sizes for OS/390 DFS Client and Server

The maximum file size that can be read by an OS/390 DFS client is based on the client chunk size. The OS/390 DFS client supports 2 Gigabyte (GB) chunks. So, if the client chunk size is 64K (typical for disk caching), then the maximum file size the OS/390 DFS client could process is 64K×2GB or (2^{47})−1 bytes. If the client chunk size is 8K (typical for memory caching), the maximum file size the OS/390 DFS client could process is 8K×2GB or (2^{44})−1 bytes. (The OS/390 DFS client chunksize is specified in the DFS client **_IOE_CM_PARMS** environment variable.)

The maximum file size that can be processed by the OS/390 DFS Server is based on the Physical File System that the data resides in. For HFS, the maximum file size is (2^{43})−1 bytes. For DCE Local File System, the maximum file size is 2GB−1 bytes. For RFS, the maximum file size is 4GB−1 bytes.

Chapter 3. Accessing Data in DFS

The DFS filespace is an extension of a machine's local file system. It allows users to access files stored on other machines. In DFS, instead of accessing only files stored on your local disk (such as those stored in the **/usr** or **/bin** directories or their equivalents), you can access all files available in the DCE namespace.

When working with DFS, you use local operating system commands to move to different directories in the filespace and to list and access files. However, DFS provides commands that support additional functionality. Among other things, these commands allow you to work with files stored on DCE Local File System and non-Local File System filesets. They also allow system administrators to set the amount of disk space allotted to each fileset.

Naming Conventions for File System Objects

In DFS, file system objects (files and directories) have names similar to object names in typical nondistributed file systems. However, because the file system is distributed, some additional elements are required to make the objects accessible across local and foreign cells.

The first element in a DFS object name is **/...** (the global namespace designation). The global namespace designation unites every cell in one uniform namespace. The second element in a DFS name is the name of a cell; for example, the cell name for a company named **abc** could be **abc.com**. The cell name is followed by the filespace designation for the cell (usually **fs**). The typical pathname associated with a file or directory in any file system completes the name of the object.

For example, suppose the user named **terry** is from the cell named **abc.com**. The user's home directory (the directory in which **terry** should initially be placed after authenticating to DCE) may be named **/.../abc.com/fs/usr/terry**. In the directory name, **/.../abc.com** indicates the global cell name; **fs** is the filespace designation in the **abc.com** cell; and **usr/terry** is the name of the user's home directory. If the file named **purchasing.memo** resides in **terry**'s home directory, the file may be stored with the full pathname **/.../abc.com/fs/usr/terry/purchasing.memo**.

Using Local Prefixes to Save Keystrokes

A cell-relative, or local, prefix, **/.:**, exists to indicate the global namespace designation and the cell name. A cell-relative name (one that uses the cell-relative prefix) can be used only when referring to files and directories in the local cell. With this prefix, the file name in the previous example can be shortened to **/.:/fs/usr/terry/purchasing.memo**.

An additional DFS-relative prefix, **/:**, is also available to signify the global namespace designation, the cell name, and the filespace designation. A DFS-relative name (one that employs the DFS-relative prefix) can also be used only when referring to files and directories in the local cell. With this prefix, the file name in the previous example can be further shortened to **/:/usr/terry/purchasing.memo**.

Your system administrator determines if these prefixes are available on your client machine. If they are, you can use them only interactively (at the command line) or in persistent storage (for example, in a shell script). However, exercise caution when using the prefixes in persistent storage; they indicate the proper pathname only when used within the local cell. For example, if a shell script intended for use outside the local cell includes a pathname to a file in the local cell, use the global name of the file, including the **/...** prefix.

Note: The examples and output in this documentation are displayed in DNS format. Use whichever format, DNS or GDS, is appropriate for your cell.

Table 3. Pathname notation shortcuts

Prefix	Replaces	Example
/.:	global namespace designation and cell name	**/.../abc.com/fs/usr/terry/purchasing.memo** becomes **/.:/fs/usr/terry/purchasing.memo**
/:	global namespace designation, cell name and filespace designation	**/.../abc.com/fs/usr/terry/purchasing.memo** becomes **/:/usr/terry/purchasing.memo**

Accessing Files and Directories

When working with files and directories in DFS, you use standard operating system commands such as **ls** and **cd** to list files and change directories. The difference is that, in DFS, you can access much more data because you are not limited to just those files and directories on your local disk. As in any file system, you can access only those files and directories for which you have permission.

In DFS, the permissions associated with a DCE Local File System file or directory can be set using DCE Access Control Lists (ACLs). Permissions are set for non-Local File System files and directories and files and directories that do not use ACLs with the standard operating system mode bits. DFS uses a specific implementation of DCE ACLs. See Chapter 5, “Using ACLs and Groups” on page 75 for important information about using ACLs to limit access to files and directories in DFS.

To access files and directories in a DCE cell, you must:

- Be authenticated to DCE

  OS/390 DFS users can be authenticated to DCE by several techniques:

  - You can issue the **dce_login** command.
  - You can be logged in as a result of executing a DCE application and being logged in by DCE (DCE single sign-on).
  - You can be logged in as a result of executing a file or directory request for a file or directory that resides in the DFS filespace (DFS single sign-on from the DFS client). For more information see “DCE Single Sign-On from the DFS client” on page 30.
- Specify correct pathnames
- Have access to the desired files and directories based on the ACLs or mode bits associated with them.

In any file system, if multiple users modify the same file at the same time, the changes last saved are the changes you see, regardless of who modified the file. When working with someone on the same files, make sure you coordinate your work so that you do not overwrite each other's changes. You can also use ACLs to limit access to your files and directories, thus preventing other users from accidentally overwriting your files.

Changing to a Different Directory

In DFS, standard operating system commands are used to change directories. Changing directories involves moving from the current working directory to a different directory. The current working directory is the directory in which you are presently located. When using the UNIX operating system with DFS, you can use the **pwd** command to display the full pathname of the current working directory on the screen.

For example, if the user **terry** is from the **abc.com** cell, **terry**'s home directory can have a name like **/.../abc.com/fs/usr/terry**. In the UNIX operating system, if **terry**'s current working directory is **terry**'s home directory, **terry** can issue the **pwd** command to display output similar to the following:

```
$ pwd

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

In the UNIX operating system, the **cd** command is used to change directories. When changing directories, the notion of a parent directory can be used to shorten the pathname required with the command. A directory's parent directory is the directory in which it resides (the directory located immediately above it in the file system hierarchy).

For example, the directory named **/.../abc.com/fs/usr/terry/public** is located beneath, or in, the directory named **/.../abc.com/fs/usr/terry**. Therefore, the directory named **terry** is the parent directory of the directory named **public**.

If **terry** is currently working in **terry**'s home directory, the user can move from that directory to the **public** directory by specifying the full pathname of the **public** directory with the **cd** command. However, because the home directory is the parent of the **public** directory, **terry** can also include just the name of the **public** directory with the command. When issued from **terry**'s home directory, the following UNIX commands are equivalent:

```
$ cd /.../abc.com/fs/usr/terry/public
$ cd public
```

In the UNIX operating system, **terry** can issue the **cd** command without a pathname to return to the home directory from the **public** directory or from any directory anywhere in the file system. When issued without a pathname argument, the **cd** command returns users to their home directories.

Listing the Contents of a Directory

Standard operating system commands are also used in DFS to list the contents of a directory. Listing a directory displays the names of all of the files and directories located in that directory. Listing a directory does not display the contents of any subdirectories (directories located in the directory being listed).

To list the contents of a directory in the UNIX operating system, enter the **ls** command with the pathname for the directory. The following example uses the **ls** command to list the files and directories located in the directory named **/.../abc.com/fs/usr/terry/public**. Enter this command:

```
$ ls /.../abc.com/fs/usr/terry/public
```

To see output like this:

```
finance.memo    misc.text      purchasing.memo
finance.text    purchasing
```

The **ls** command can also be issued with the **-F** option to indicate, among other things, each subdirectory in a directory. If the **-F** option is included, a / (slash) is displayed after the name of each subdirectory. For example, enter the same command with the **-F** option:

```
$ ls -F /.../abc.com/fs/usr/terry/public
```

To see output like this:

```
finance.memo    misc.text      purchasing.memo
finance.text    purchasing/
```

Note that **purchasing** now has a slash after its name in the output to identify it as a directory. Also note that in both examples the contents of the **purchasing** directory are not displayed when its parent directory, **public**, is listed.

You can also use the **cd** command to change to a directory and then issue the **ls** command to list the contents of the directory without specifying the directory's name. When issued without a pathname argument, the **ls** command lists the contents of the current working directory.

Using Name Prefixes in Commands

When issuing commands in DFS, you can also use **/.:** (the cell-relative prefix) and **/:** (the DFS-relative prefix) if your system administrator for your cell has enabled them. For example, the following commands use the two abbreviations to execute the UNIX **cd** and **ls** commands displayed in the previous sections. When issued from the local cell, the following three **cd** commands are equivalent:

```
$ cd /.../abc.com/fs/usr/terry/public

$ cd /.:/fs/usr/terry/public

$ cd /:/usr/terry/public
```

Similarly, when issued from the local cell, the following three **ls** commands are equivalent:

```
$ ls /.../abc.com/fs/usr/terry/public

$ ls /.:/fs/usr/terry/public

$ ls /:/usr/terry/public
```

Locating Files and Directories

You do not need to know which File Server machine stores your files or any other files you want to access. When given a pathname, the Cache Manager on your machine automatically finds the machine that houses the file and retrieves the appropriate data. However, it is sometimes useful to know the location of a particular file. For example, if a File Server machine becomes unavailable and files you are editing are stored on that server, you must wait until the server returns to service to save your files.

In addition, when using DCE Local File System, your system administrator can move filesets to different File Server machines. Moving filesets allows your administrator to use system disk storage to balance the load on your system across different machines more efficiently. You may sometimes need to check the current location of a fileset to verify the status of the server machine on which it is stored in the event of a server machine outage.

Use the **cm whereis** command to learn the location of the fileset that contains a file or directory. This command produces a list of the File Server machines that store the fileset containing a file or directory, as well as the cell in which the fileset resides and the fileset's name.

A list of more than one machine means that replicas of the fileset that contains the file or directory are stored on different machines. This usually indicates a demand for the fileset's contents. For example, binary files such as those for text editors are often replicated on different machines; if a machine housing the files becomes unavailable, other copies of the files can still be accessed on other machines. Because they are rarely in demand by more than one user, user filesets are seldom replicated.

Using the cm whereis Command

Use the **cm whereis** command to determine the location of a file or directory:

```
$ cm whereis [-path {filename | directory_name}...]
```

The **-path** option is the name of each file or directory whose location you want to know; the **...** (ellipsis) indicates that multiple file or directory names can be specified. If a complete pathname is not specified for a file or directory, the file or directory is assumed to reside in the current working directory. Omit this option and its arguments to learn the location of the current working directory.

The following example displays the name of the machine on which the home directory for the user named **terry** is located. The output from the command indicates that **terry**'s home directory resides in the cell named **abc.com**, on the fileset named **user.terry**, on the File Server machine named **fs2**. If you enter:

```
$ cm whereis /.../abc.com/fs/usr/terry
```

The output will look like this:

```
The file '/.../abc.com/fs/usr/terry' resides in the cell
'abc.com', in fileset 'user.terry', on host fs2.abc.com.
```

The following example illustrates the use of the **cm whereis** command to learn the location of more than one file or directory. The output from the command indicates that the **plans** file, which is located in the current working directory, resides on the File Server machine named **fs2**, and the file named **strategy**, which is located in the home directory of the user named **pat**, resides on the File Server machine named **fs3**. The filesets that house the two files reside in the cell named **abc.com**; their respective names are **user.terry** and **user.pat**. If you enter:

```
$ cm whereis plans /.../abc.com/fs/usr/pat/strategy
```

The output will look like this:

```
The file 'plans' resides in the cell 'abc.com', in fileset
'user.terry', on host fs2.abc.com.

The file '/.../abc.com/fs/usr/pat/strategy' resides in the cell
'abc.com', in fileset 'user.pat', on host fs3.abc.com.
```

Listing Fileset Quota Information

In DFS, your files are usually stored together on one fileset on the disk of a File Server machine. To divide your cell's available disk space as efficiently as possible, your system administrator imposes a size limit, or quota, on each fileset. By default, every newly created DCE Local File System fileset has a maximum quota of 5000 kilobytes (5000 units of 1024 bytes each).

You can use the **fts lsquota** command to determine the quota of your fileset. When entering the **fts lsquota** command, you can indicate the fileset whose quota you want to display either directly, by specifying the name (or fileset ID number) of the fileset, or indirectly, by specifying the name of a file or directory located in the fileset. You can also use the command to display quota information about multiple filesets by specifying either multiple fileset names or multiple file or directory names.

The information displayed is different for DCE Local File System and non-Local File System filesets. The **fts lsquota** command displays the following information about a DCE Local File System fileset:

- The name of the fileset. For users, this is often **user.***username*; for example, **user.terry** or **user.pat**.
- The size of the quota for the fileset, expressed as a number of kilobytes. For example, the number 1024 indicates the quota is 1 megabyte (1024 kilobytes).

- The number of kilobytes currently in use on the fileset.
- The percentage of the quota currently in use on the fileset.
- The percentage of available disk space currently in use on the aggregate on which the fileset resides. Remember, a DCE Local File System aggregate holds not only your fileset, but the filesets of other users as well.
- The total number of kilobytes and number of kilobytes currently in use on the aggregate.
- An `LFS` indicator to show that the fileset is stored on a DCE Local File System aggregate.

The command displays the following information about a non-Local File System fileset:

- The name of the fileset.
- Zeros for the size, usage, and percentage used of the non-Local File System fileset. When using the **fts lsquota** command to determine the size of a non-Local File System fileset, ignore the quota, usage, and percentage values displayed for the fileset; they are always 0 (zeros). Consult the quota, usage, and percentage values displayed for the partition on which the non-Local File System fileset resides to determine the corresponding values for the fileset.
- The percentage of kilobytes in use, the number of kilobytes in use, and the number of kilobytes available on the non-Local File System aggregate (partition).
- A `non-LFS` indicator to show that the fileset is stored on a non-Local File System partition.

Files stored on a non-Local File System partition are still considered to be stored on a fileset. However, the fileset in this case is equal in size to the disk partition on which the fileset resides. The quota of the fileset, therefore, is equal to the size of the disk partition. In OS/390 DFS, you can use the **fts lsquota** command to determine the size of the fileset. (Except for Record File System (RFS) filesets, which give you dummy values.)

You need to periodically check the quota of your fileset to verify that you continue to have adequate space. If you are close to exceeding your fileset quota, you may not be able to save changes to a file. Similarly, you are not able to save a file if the aggregate that stores your fileset is full, even if you are not close to exceeding your quota. (An aggregate can be full even if one or more of the filesets it contains has quota remaining.) Check your fileset quota if you have problems saving files; if necessary, remove some files to create free disk space, or ask your system administrator to increase your quota. (The **fts setquota** command used to increase the quota of a DCE Local File System fileset cannot be used to increase the quota of a non-Local File System fileset.)

You also need to check your fileset quota before copying or moving large files to your fileset. A copy or move operation fails if it causes your fileset to exceed its quota. This is true of all filesets, so check the quota of any fileset to which you want to copy or move data before attempting the operation.

Note: The **fts lsquota** command can also be used to determine the name of a fileset from its fileset ID number. Simply enter the ID number with the command's **-fileset** option.

Using the fts lsquota Command

Enter the **fts lsquota** command to display quota and usage information about one or more filesets:

```
$ fts lsquota [{-path {filename | directory_name}... | -fileset {name | ID}...}]
```

The **-path** option is the name of a file or directory in each fileset about which information is to be displayed. Include the names of files or directories from different filesets if desired. It is not necessary to name more than one file or directory from the same fileset. Use this option or use **-fileset**; omit both options to display information about the fileset containing the current working directory.

The **-fileset** option is the complete name or fileset ID number of each fileset about which information is to be displayed. Use this option or use **-path**; omit both options to display information about the fileset containing the current working directory.

In the following example, the user named **terry** issues the **fts lsquota** command from the user's home directory to check the quota for the fileset that contains the directory. The output shows that the DCE Local File System fileset named **user.terry**, which contains **terry**'s home directory, has a quota of 15,000 kilobytes; 5071 kilobytes, or 34% of **terry**'s quota, are currently in use. It also shows that 86% (84,538 kilobytes out of an available 98,300 kilobytes) of the aggregate on which **terry**'s home directory and fileset are located is in use. Enter this command:

```
$ fts lsquota
```

to see output like this:

```
Fileset Name    Quota   Used   % Used   Aggregate
user.terry      15000   5071      34%     86% = 84538/98300 (LFS)
```

The following example displays quota and usage information about an HFS fileset, **user.jlw**, which contains the directory named **/.../abc.com/fs/usr/jlw**.

Because the partition on which the HFS fileset resides can contain only the **user.jlw** fileset, the quota, usage, and percentage information displayed for the HFS aggregate (partition) apply for the fileset. Therefore, the quota for **user.jlw** is 10,000 kilobytes, the number of kilobytes in use on the fileset is 8448, and the percentage of the fileset in use is 84%. Enter this command:

```
$ fts lsquota /.../abc.com/fs/usr/jlw
```

to see output like this:

```
Fileset Name    Quota   Used   % Used   Aggregate
user.jlw       100000   8448      84%     84% = 8448/10000 (non-LFS)
```

Checking File Server Machine Status

File Server machines in your cell sometimes become unavailable due to hardware problems, software problems, or routine maintenance. If you experience problems using binary files or saving files you have edited, you can use the **cm statservers** command to check the statuses of the File Server machines that house the files. When a File Server machine is unavailable, you cannot access files stored on that machine or save any changes you made to locally cached versions of files that reside on that machine until the machine returns to service. Your Cache Manager keeps copies of files you cannot save in the local cache until the File Server machine is again available.

If a File Server machine is unavailable, your Cache Manager can still have copies of files from the File Server machine cached locally, either in memory or on disk. You can continue to work with the copies of these files on your local machine, but you cannot save them back to the File Server machine on which they reside until it returns to service.

The **cm statservers** command lists the names of unresponsive File Server machines. A File Server machine is classified as unresponsive if it meets the following two conditions:

- The Cache Manager cannot save cached data to the machine.
- The machine does not respond to the Cache Manager's status probes or requests for data.

The Cache Manager probes all machines that house source versions of data the Cache Manager has cached or is attempting to cache; it does not probe all File Server machines in a cell. This command is concerned only with File Server machines that do not respond to the Cache Manager on your client machine.

You can use the **cm statservers** command to check the statuses of File Server machines in your local cell, in a specific foreign cell, or in all cells the Cache Manager has contacted. If all of the servers it probes in the indicated cells are running, the Cache Manager displays the message `All servers are running`. Otherwise, it displays the message `These servers are still down`, followed by a list of the machines that are presently unresponsive.

This command can take some time to complete if many of the machines checked fail to respond to the Cache Manager's probes. For this reason, it is often executed in the background by specifying the & (ampersand) at the end of the command line. (The ampersand is used to execute a command in the background in the UNIX operating system; use it or the equivalent approach for your operating system.) If the command is not run in the background and it is taking a long time to complete, you can safely terminate its execution by entering the interrupt signal (<**Ctrl-c**> or the appropriate signal for your operating system).

Important Note to Users

In an OS/390 DFS environment, the notation <**Alt-c**>**c** followed by pressing the Enter key is used to indicate an interrupt signal.

You can also execute the command with the **-fast** option to list those File Server machines that are currently unresponsive. If the **-fast** option is used, the Cache Manager does not attempt to determine the statuses of the machines at the present time. It simply provides the names of machines that did not respond to its most recent probes.

If you are uncertain about whether your files are affected by a machine outage, use the **cm whereis** command to determine where the fileset housing the files is stored. If the software for an application program (a text editor, for example) is stored on only a single File Server machine and that machine is currently unavailable, you cannot initialize the program; you must wait until the machine returns to service to use the application.

Using the cm statservers Command

Enter the **cm statservers** command to determine the status of each machine the Cache Manager must contact to write data to files or to obtain data from files:

```
$ cm statservers [{-cell cellname | -all}][—fast]
```

The **-cell** *cellname* option is the name of the foreign cell whose File Server machines the Cache Manager is to check. The Cache Manager determines the statuses of only those machines it must contact in the specified cell. Use this option or use **-all**; omit both options to check the statuses of only those machines the Cache Manager must contact in the local cell.

The **-all** option specifies that the Cache Manager is to check the statuses of all File Server machines it must contact, regardless of the cells in which the machines reside. Use this option or use **-cell**; omit both options to check the statuses of only those machines the Cache Manager must contact in the local cell.

The **-fast** option directs the Cache Manager to display the names of File Server machines it has recently contacted, without probing the machines to determine whether they are currently available. This option can be combined with the **-cell** or **-all** option, or it can be used alone.

The following example checks the status of each File Server machine the Cache Manager must contact, regardless of the cell in which a machine is located. The & (ampersand) is used to run the command in the background.

```
$ cm statservers -all &

These servers are still down: fs1.abc.com fs3.abc.com.
```

Chapter 4. DFS Configuration Issues

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.

This chapter also provides summary information about the following DFS configuration issues:

- Choosing DFS machine roles,
- DFS server and client configuration issues,
- Setting up DCE Local File System filesets,
- Understanding DFS data access management,
- Understanding DFS data communication security,
- Understanding the DFS distributed database technology.

Subsequent chapters provide specific details about managing DFS server machines, processes, filesets, and files.

Important Note to Users

This chapter discusses some processes and commands that are not available in OS/390 DFS. They are:

- OS/390 cannot act in a binary distribution role on either client or server.
- The Private File Server Machine
- The **ioedfsd** process is also unique in OS/390 in that it combines the functions of the traditional Cache Manager **dfsd** and **dfsbind** processes. In OS/390, references to the **dfsd** or **dfsbind** process should be taken to mean the **ioedfsd** process.

Choosing DFS Machine Roles

DFS server and client machines can run the following processes:

1. The BOS Server is used to monitor other processes.
2. The Fileset, Fileset Location, and Replication Servers are used to manipulate DFS filesets and their replicas.
3. The Backup Server is used to contact the Backup Database.
4. The **butc** process is used to back up file system data to tape.
5. The **dfsd** process is used to initialize the Cache Manager on a client machine. In OS/390, the **dfsd** process exists as part of the **DFSCM** address space (**ioedfsd**) and is invoked by the **ioecmini** process during initialization.

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 must also be created in the Registry Database for each server machine. The following sections assume that these requirements have been satisfied prior to configuring a machine as a DFS server or client machine.

The System Administrator determines, at installation, which processes are to run on which machines. A machine's role is determined by the types of processes it runs. The information in the following sections detail 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 6, "Using Administrative Lists and Keytab Files" on page 103 provides detailed information about the procedures used to create and maintain administrative lists. See Chapter 19, "Distributed File Service Commands" on page 401 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.

The BOS Server on each server machine is set to stop and immediately restart all DFS processes (including itself) on the machine. Generally, in non-OS/390 DFS systems, if a restart time is not specified, it defaults to restart each week at 4:00 a.m. on Sunday. In OS/390 DFS systems, the default restart time is **never**, meaning that a server restart is not performed. Restart times can, however, be specified. See the description of the **bos setrestart** command in "bos setrestart" on page 522 for more information on setting restart times for the BOS server.

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

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

Overview of DFS Machine Roles

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

- **System Control machine** — A single machine acts as the System Control machine for a domain, updating the other machines in the domain with identical versions of common configuration files such as administrative lists.
- **Binary Distribution machine**

  Important Note to Users

  OS/390 DFS cannot act in a binary distribution role on either client or server.

- **File Server machine** — A File Server machine runs the basic set of processes necessary for storing and exporting DCE Local File System and non-Local File System 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 multi-user workstation.

  Note: In OS/390, the Cache Manager services are invoked from OMVS when any directory or file with the pathname prefix **...** is referenced.

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: The System Control machine is 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 “DFS Distributed Database Technology” on page 69 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, start or stop the appropriate processes. Machines roles are not mutually exclusive; any server machine can assume multiple server machine roles, any server machine can be configured as a client machine, and any client machine can be configured as a server machine.

System Control Machines

The System Control machine in a domain stores and distributes system configuration information, such as administrative lists, shared by all DFS server machines in the domain. Configure the first server machine for any new domain as the System Control machine for that domain. It can then be used to distribute the administrative lists for that domain from its **/opt/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 common configuration files from the Update Server.
- A BOS Server (**bosserver** process) — (See “Choosing DFS Machine Roles” on page 43 for more information about the BOS Server).

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

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

Binary Distribution Machines

Important Note to Users

OS/390 DFS cannot act in a binary distribution role on either client or server.

File Server Machines

A File Server machine is used to store and export DCE Local File System or non-Local File System 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).
- The **upclient** process — retrieves configuration files from the System Control machine.
- A BOS Server (**bosserver** process). (See "Choosing DFS Machine Roles" on page 43 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 Local File System fileset, and they can effectively change the UNIX permissions on all of the files and directories in each exported non-Local File System 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.

While 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 the user accesses the file or directory:

- When accessing a file or directory by 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: In some DFS systems, the DCE identity **root** effectively has **root** privileges for data in all exported non-Local File System 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.

  In OS/390 systems, remote clients with the DCE identity **root**, do not necessarily have root privileges on OS/390 exported non-Local File System filesets. A DFS client's DCE identity coming into an OS/390 system is mapped to a local OS/390 identity (see “Mapping DCE User IDs to OS/390 User IDs” on page 174). The **UID** of the OS/390 identity is determined by RACF and may or may not be **root**.

- When accessing a file or directory by 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 Local File System 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-Local File System fileset can always be accessed by a local pathname because a non-Local File System fileset must always be mounted locally, as a file system on its File Server machine. A file or directory in a DCE Local File System fileset can be accessed by 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 first changing the protections. Being authenticated as DCE **root** lets you perform any operation on a file or directory in an exported non-Local File System fileset immediately on some systems.

The File Exporter also manages the distribution of tokens to clients. It maintains an inventory of outstanding tokens, including the clients to which it has granted tokens, the data for which it has granted those tokens, and the type of each token it has granted. (A token's type dictates the operations that the client holding the token can perform on the data to which the token applies.) (See "Data Access Management in DFS" on page 64 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 "fxd" on page 722 for complete information about the **fxd** 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 automatically creates replicas 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. 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 Local File System or non-Local File System fileset from the machine. (See Chapter 10, "Making Filesets and Aggregates Available" on page 147 for more information about creating server entries.)

A client machine can also be configured as a Private File Server machine to export data to the global namespace. (See "Exporting Data from a Client Machine (Private File Server Machine)" on page 50 for more information about configuring a client machine to export data.)

Fileset Database Machines

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

Each Fileset Database machine runs the following processes:

- A Fileset Location Server (**flserver**) process.
- The **upclient** process — retrieves configuration files from the System Control machine.
- A BOS Server (**bosserver** process). (See "Choosing DFS Machine Roles" on page 43 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 the user 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 10, “Making Filesets and Aggregates Available” on page 147 for more information about creating server entries in the FLDB).

Backup Database Machines

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

Each Backup Database machine runs the following processes:

- A Backup Server (**bakserver** process).
- The **upclient** process — retrieves configuration files from the System Control machine.
- A BOS Server (**bosserver** process). (See “Choosing DFS Machine Roles” on page 43 for 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 14, “Configuring the Backup System” on page 277 for more information on configuring and using Tape Coordinator machines.)

DFS Client Machines

A DFS client machine serves primarily as a single or multi-user 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.

Important Note to Users

The **ioedfsd** process is also unique in OS/390 in that it combines the functions of the traditional Cache Manager **dfsd** and **dfsbind** processes. In OS/390, references to the **dfsd** or **dfsbind** process should be taken to mean the **ioedfsd** process.

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

When the Cache Manager receives the requested data, it stores the data in its local cache, which is an area reserved for data storage on disk or in memory on the client machine. It then passes the data to the application program. The Cache Manager also stores tokens it receives from the File Exporter on the File Server machine.

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

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

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

The **dfsbind** process, in addition to its role on File Server machines, is used by the Cache Managers on client machines to help with the resolution of DCE pathnames. It also obtains authentication information about users that Cache Managers require for RPC bindings to File Server machines. (See “Client Machine Processes and Files in OS/390” on page 52 for more information about the Cache Manager and the **dfsd** and **dfsbind** processes.)

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

Important Note to Users

The Private File Server machine is not available in OS/390 DFS.

Summary of DFS Machine Roles

Table 4 on page 51 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.

Any server machine can be configured to perform other server machine roles. Also, a server machine can be configured as a client machine, and vice versa. For a machine to fill an additional role, it must run the processes listed for that role in the third column of the table.

Table 4. Summary of DFS Machine Roles

Machine Role	Purpose	Process	Recommendations
System Control machine	Distribute common configuration files for a domain	**bosserver** **upserver**[1]	Use an OS/390 DFS machine as the System Control machine for a domain.
Binary Distribution machine	Distribute system binary files for its CPU/OS type	**bosserver** **upserver**[2] **upclient**[2]	OS/390 DFS cannot act in a binary distribution role on either client or server.
File Server machine	Export and store DCE Local File System and non-Local File System data	**bosserver** **ftserver** **fxd** **dfsbind**[3] **repserver** **upclient**[1]	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	Store the Fileset Location Database (FLDB)	**bosserver** **flserver** **upclient**[1]	Configure three Fileset Database machines.
Backup Database machine	Store the Backup Database	**bosserver** **bakserver** **upclient**[1]	Configure three Backup Database machines.
DFS Client machine	Serve as a single-user or multi-user workstation; access files for application programs	**dfsd**[4] **dfsbind**[3] **ioecmini**[4] **ioelogin**[4]	
DFS Gateway machine	Maps user identity for NFS requests to DCE identity	**dfsgwd**[3]	See Appendix D, "DFS/NFS Secure Gateway" on page 805.

1 The notation **upserver**[1] and **upclient**[1] denote the Update Server that distributes common configuration files from a System Control machine.

2 The notation **upserver**[2] and **upclient**[2] denote the Update Server that distributes binary files from a Binary Distribution machine.

3 This process is not supported in OS/390 DFS on the OS/390 operating system but the equivalent function is provided.

4 In OS/390, the **dfsd** process exists as the **ioedfsd** process in the **DFSCM** address space with the **ioecmini** and **ioelogin** processes. The **ioecmini** process is an OS/390 specific process that is invoked as a physical file system during OMVS initialization. This process initializes the Cache Manager in OS/390 and invokes the **ioedfsd** process. The **ioelogin** process is an OS/390 specific process that performs DCE login processing; one or more **ioelogin** processes may exist in the **DFSCM** address space.

DFS Server and Client Configuration Issues

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

Server Machine Processes and Files

You should combine machine roles for the machines in your cell and domains. For example, you may wish 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 automatically replicated to other machines 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.

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

- The **/opt/dfsglobal/bin** directory contains DFS binaries that are appropriate for the machine's CPU/OS type. The binary files are for command suites and programs (such as the **scout** program).
- The **/opt/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** function. If the machine is running the Fileset Location Server or the Backup Server, this directory also contains the FLDB or the Backup Database.
- The **/opt/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.

Client Machine Processes and Files in OS/390

The OS/390 DFS client is a physical file system run as **DFSCM** in a colony address space. (For more information on the colony address space see the *OS/390 UNIX Sytem Services File System Interface Reference*, SC28-1909.) The **DFSCM** address space is started during OS/390 UNIX initialization.

Note: The traditional DFS client machine **dfsd** process functions and **dfsbind** process functions are merged as part of **DFSCM**. The **ioecmini**, **ioedfsd**, and **ioelogin** processes run in the **DFSCM** address space (see Figure 2 on page 53). The **dfsbind** process does not exist separately in OS/390 DFS. In OS/390 DFS documentation, references to **dfsd** and **dfsbind** should be interpreted as referring to the equivalent function that exists as part of the **DFSCM** address space.

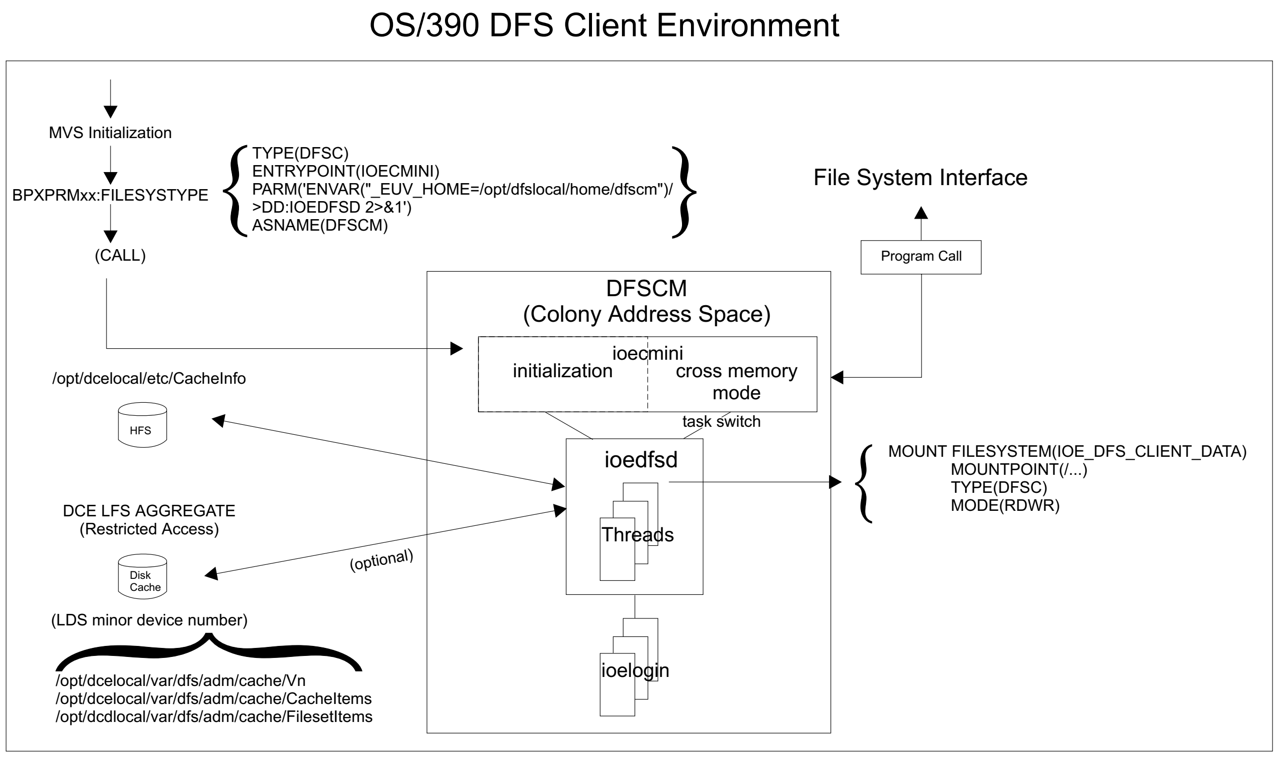


Figure 2. DFSCM Address Space

The DFS filespace is an extension of the Local File System. Users can access files stored on other OS/390 or non-OS/390 machines. Instead of accessing only files available on your local disks (such as those stored in the **/usr** or **/bin** directories or their equivalents), you can access all files available in the DCE namespace.

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.

DFSCM initializes the Cache Manager on a client machine by transferring configuration information to OS/390 UNIX. It also mounts the root of the global namespace (**/...**).

Note: OS/390 DFS uses a single Linear Data Set (LDS) to contain all the required DFS client caching related files.

In addition, **DFSCM** starts several processes that run in the colony address space. The **ioedfsd** process includes one or more maintenance threads that perform routine maintenance tasks such as garbage collection; background threads that improve performance by performing delayed writing of updated data; token threads that respond to token revocation requests from File Exporters; and, on OS/390, I/O threads that move data between disk and memory.

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

DFSCM also acquires authentication information from the Security Server. Authentication information must be included in the 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 Chapter 8, "Starting and Stopping the DFS Cache Manager in OS/390 DFS" on page 125 for information about options available for controlling **DFSCM**.

Client Machine Files: 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 **DFSCM** reinitializes the Cache Manager. OS/390 DCE must be running before **DFSCM** can complete initialization. Any files that are needed during reboot and prior to the start of **DFSCM** must reside on the local disk. Following is a list of recommended DFS files to store on a local disk.

- **/opt/dcelocal/etc/CacheInfo** is a file that specifies three aspects of Cache Manager configuration separated by colons:
  1. Where the DCE (or DFS) global namespace is mounted (e.g **/...**)

     Note: Mounting the global namespace at other than **/...** is not supported.
  2. The minor device number of the DCE Local File System aggregate that contains the DFS client cache related files
  3. The cache size in 1K blocks (e.g. 45000)

  If this file does not exist, the following defaults are taken, (**/...:998:8000**). You must either be running with a memory cache (see "DFS Client ENVAR File" on page 274) or you must have a disk cache DCE Local File System aggregate with a minor device number of **998** (see "Disk Cache Configuration" on page 257).
- A single DCE Local File System aggregate whose minor device number is specified as the second entry of the CacheInfo file that contains the cache related files:
  - **/opt/dcelocal/var/dfs/adm/cache/V***n* files where data chunks are held.
  - **/opt/dcelocal/var/dfs/adm/cache/CacheItems** file where information is recorded about each V*n* file.
  - **/opt/dcelocal/var/dfs/adm/cache/FilesetItems** file containing fileset to mountpoint mappings.

Note: The following directory and files are NOT required for the OS/390 DFS client:

- **/opt/dcelocal/var/dfs/adm/cache**
- **/opt/dfsglobal/bin/dfsbind**
- **/opt/dfsglobal/bin/dfsd**

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 wish to keep the binaries for a text editor, such as **ed** or **vi**, on a local disk for use during outages.

Multihomed Server Configuration Issues

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

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

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

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

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

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

Cache Manager preferences are explained in detail in Chapter 13, “Configuring the Cache Manager” on page 251.

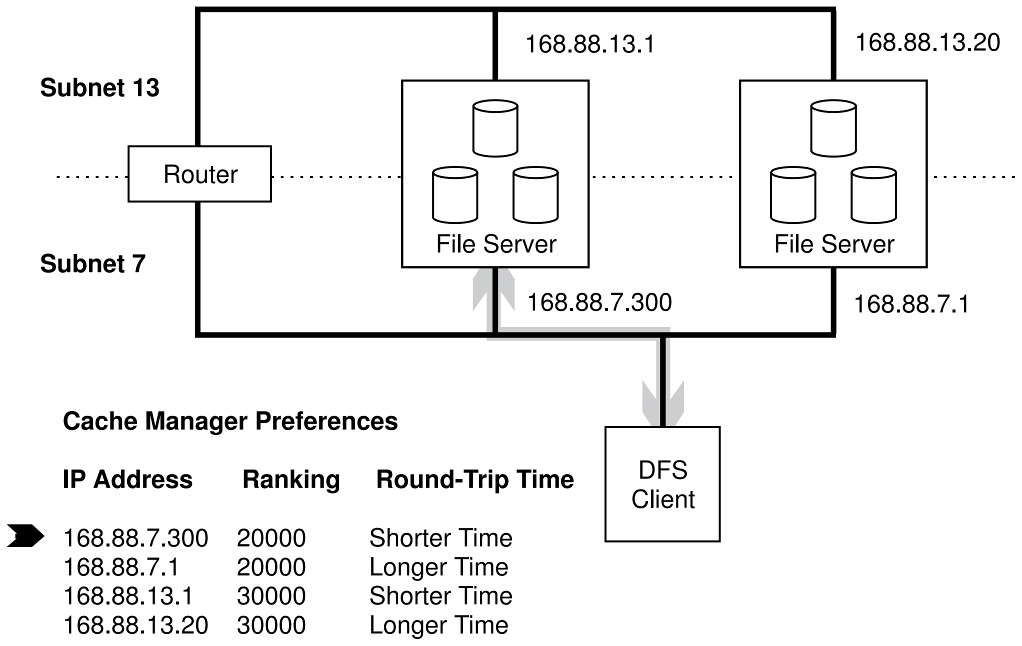
For example, a server on the same machine as the Cache Manager receives a rank of 5000, while a server on the same subnetwork receives a rank of 20,000. The entry with the lowest ranking value has the highest preference. Thus, a server with a preference value of 5000 will be chosen before a server with 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 14, “Configuring the Backup System” on page 277.

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 3.

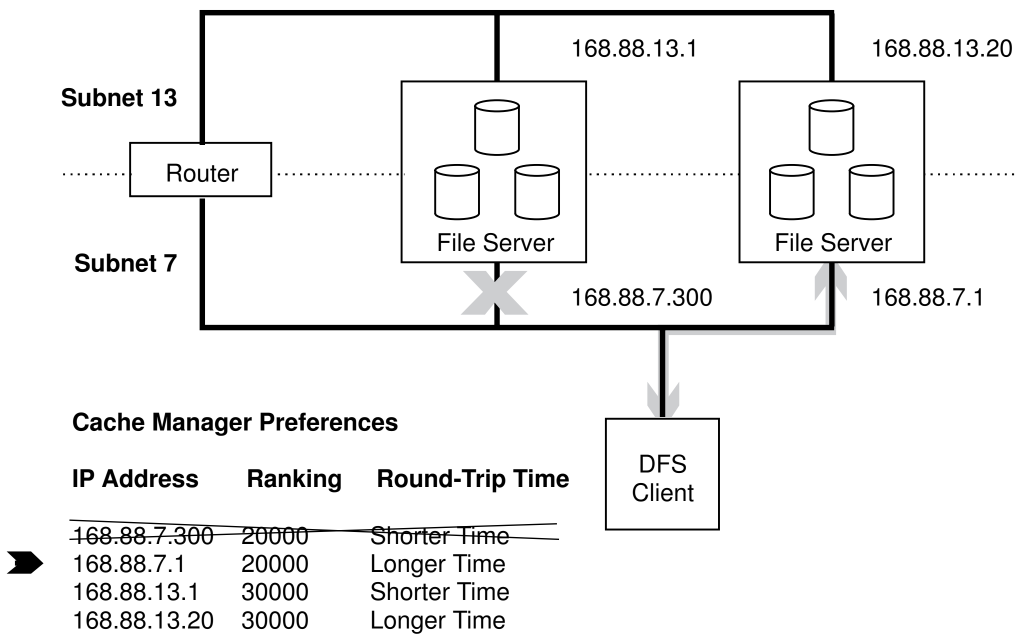


Cache Manager Preferences

IP Address	Ranking	Round-Trip Time
168.88.7.300	20000	Shorter Time
168.88.7.1	20000	Longer Time
168.88.13.1	30000	Shorter Time
168.88.13.20	30000	Longer Time

Figure 3. 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 4 on page 57. 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.

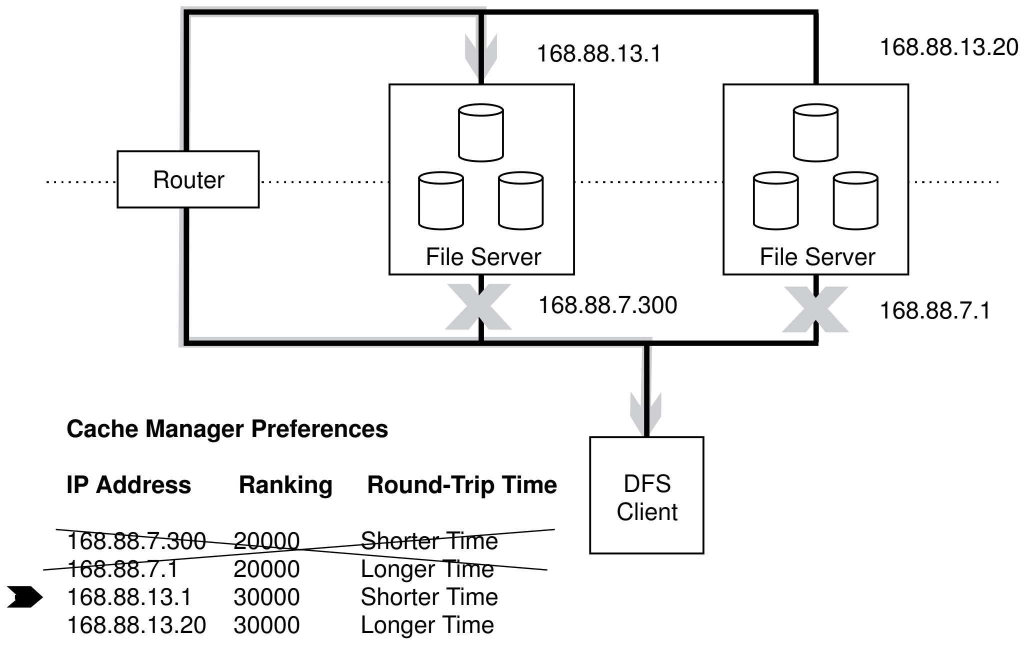


Cache Manager Preferences

	IP Address	Ranking	Round-Trip Time
	~~168.88.7.300~~	~~20000~~	~~Shorter Time~~
➤	168.88.7.1	20000	Longer Time
	168.88.13.1	30000	Shorter Time
	168.88.13.20	30000	Longer Time

Figure 4. 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 in Figure 5. 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.



Cache Manager Preferences

	IP Address	Ranking	Round-Trip Time
	~~168.88.7.300~~	~~20000~~	~~Shorter Time~~
	~~168.88.7.1~~	~~20000~~	~~Longer Time~~
➤	168.88.13.1	30000	Shorter Time
	168.88.13.20	30000	Longer Time

Figure 5. Cache Manager Again Losing Connection and Contacting File Server Address in Another Subnet

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

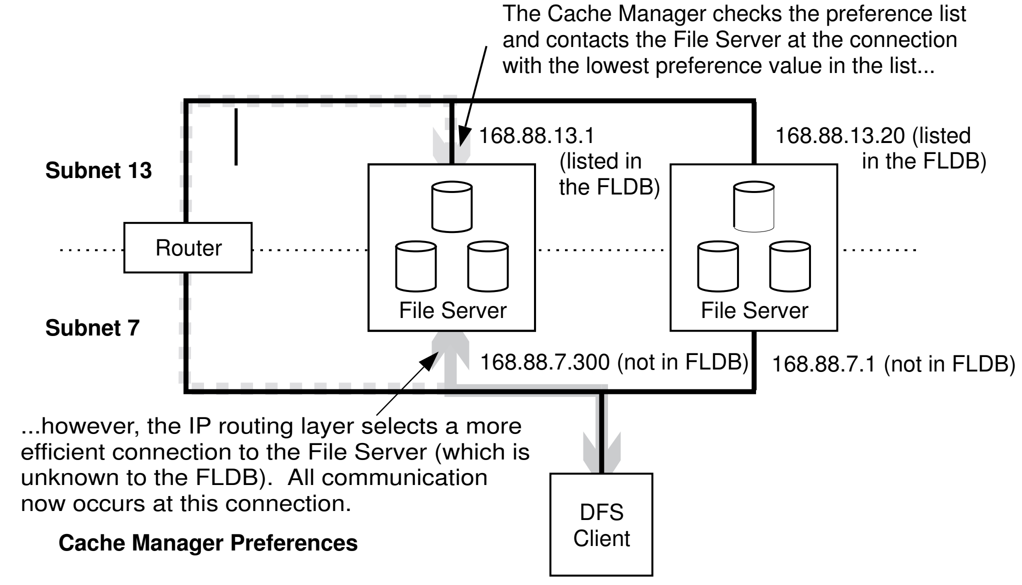
Tasks to Administer Multihomed Servers: The following are tasks to configure a multihomed File Server environment:

- You must add each host name or IP address for each File Server to the Fileset Location Database (FLDB). The initial entry, along with one address for that entry, is created using the **fts crserverentry** command. Additional addresses can be added or deleted from the entry by using the **fts edserverentry** command. See Chapter 10, “Making Filesets and Aggregates Available” on page 147 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 13, “Configuring the Cache Manager” on page 251 explains how to both create such a file and ensure that it is loaded each time the Cache Manager is initialized.

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

Should the IP layer select a different server address, this connection becomes the connection used by the Cache Manager, regardless of its preference rank or whether or not it is one of the addresses listed in the FLDB for a given server. This scenario is shown in Figure 6.


The Cache Manager checks the preference list and contacts the File Server at the connection with the lowest preference value in the list...
168.88.13.1 (listed in the FLDB)
168.88.13.20 (listed in the FLDB)
Subnet 13
Router
File Server
File Server
Subnet 7
168.88.7.300 (not in FLDB)
168.88.7.1 (not in FLDB)
...however, the IP routing layer selects a more efficient connection to the File Server (which is unknown to the FLDB). All communication now occurs at this connection.
DFS Client

Cache Manager Preferences

	IP Address	Ranking	Round Trip Time
➤	168.88.13.1	30000	Shorter Time
	168.88.13.20	30000	Longer Time

Figure 6. An Example of the IP Layer Overriding the Cache Manager's Preference

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

Creating Additional Default Entries in the Routing Table: The preference list provides each Cache Manager with a list of known connections to various File Servers and FL Servers. This list allows the Cache Manager to select alternative connections to communicate with the appropriate servers should a network or server fault make a particular connection unavailable. Similarly, each File Server or FL Server that has multiple connections to the network should have multiple default entries in its routing table that define the various routers available to that server. Thus, if a network fault makes a particular router unavailable, that server has additional router choices that would allow it to reply to Cache Manager requests. Refer to your operating system documentation for information concerning adding default entries to a server's routing table.

Setting Up Filesets

DCE Local File System filesets are created with the **fts create** command. Non-Local File System 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 Local File System and non-Local File System filesets are created with the **fts crmount** command.

The following sections discuss setting up a cells 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. For complete information about creating and mounting filesets see Chapter 10, “Making Filesets and Aggregates Available” on page 147 and see “fts” on page 608 for complete information about the **fts** command.

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 Local File System fileset or it can be a non-Local File System fileset. However, it must be a DCE Local File System fileset if functionality such as replication is to be available in the cell.

Note: On OS/390, Record File System (RFS) is not supported for **root.dfs**.

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

```
$ fts create root.dfs -server machine -aggregate name
```

Once the root fileset is created, the **root.dfs** fileset automatically 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
```

Note: In OS/390, there may be a delay before the **.rw** mount point is available from OMVS. Issue the **cm checkfilesets** command to expedite the availability of the **.rw** mount point.

Once these steps are complete, you can replicate **root.dfs**. Replication is then available for DCE Local File System 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 10, "Making Filesets and Aggregates Available" on page 147 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**.

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 nine-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, so 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 do the following.

- 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**.

(See "Fileset Names" on page 152 for more information on additional rules for naming filesets).

Setting Up Binary and Configuration Filesets

Important Note to Users

OS/390 DFS cannot act in a binary distribution role on either client or server.

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 5 lists the organization and names of the filesets in this example.

Table 5. Examples of Fileset Names and Mount Points for User Data

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

Moving Data from Non-Local File System Directories to DCE Local File System Directories

The guidelines in the previous section 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-Local File System fileset mounted in the global namespace can continue to use that fileset. However, if you choose to create and mount a DCE Local File System 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 Local File System 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-Local File System fileset mounted at **/.../abc.com/fs/usr/terry**. You would perform the following tasks:

1. 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).
2. Remove the **/.../abc.com/fs/usr/terry** directory and its contents.
3. Create and mount the user's DCE Local File System fileset.
4. Move the user's data from the temporary location into the new DCE Local File System fileset mounted at **/.../abc.com/fs/usr/terry**.

When these steps are complete, the user can access the data as before.

Replicating DCE Local File System Filesets

You replicate DCE Local File System 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, if a read-only fileset that resides at the same site as its read/write fileset (replicated using Release Replication) becomes unavailable, all other read-only versions of that fileset become unavailable after a configurable amount of time. Similarly, if a read/write fileset (replicated using Scheduled Replication) becomes unavailable, all read-only versions of the fileset become unavailable after a configurable amount of time. (See Chapter 10, “Making Filesets and Aggregates Available” on page 147 for detailed information on the availability of read-only filesets.)

Replicate only those DCE Local File System 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 heavily used. For example, binary files for text editors or other popular 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, lessening 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 address. You can set or change the Cache Manager preferences to suit your needs. (See Chapter 13, “Configuring the Cache Manager” on page 251 for more information about the Cache Manager and File Server machine preferences.

Using the @sys and @host Variables

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

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

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

On a machine running OS/390:

```
$ cm sysname

Current sysname is 's390_os390'

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

$ pwd

/.../abc.com/fs/s390_os390
```

On a machine running AIX® 4.1:

```
$ cm sysname

Current sysname is 'rs_aix41'

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

$ pwd

/.../abc.com/fs/rs_aix41
```

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

A link that includes the **@sys** variable can be created on each client machine. The Cache Manager on each machine interprets the **@sys** variable, so each machine accesses the binary file for its system type from the global namespace. Symbolic links that include the **@sys** variable are commonly used to access binary files for programs such as **make** and **emacs**.

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

On a machine running OS/390:

```
$ 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/s390_os390/usr/local
```

On a machine running AIX 4.1:

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

$ ls -l /usr/local

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

$ cd /usr/local

$ pwd

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

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

(See “cm sysname” on page 582 for more information about the **cm sysname** command.)

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

For example, suppose two machines, **fs1.abc.com** and **fs2.abc.com**, use two different, machine-specific versions of an initialization file for an application that they start following a reboot. The name of the initialization file is **start**. The file **start** can be stored in DFS and accessed on a machine-specific basis by 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 as follows:

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

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

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

Data Access Management in DFS

All access to data and metadata on a File Server machine is managed by the File Exporter. Clients contact the File Exporter when they wish 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 until it can service it or just refuses to grant it. A client that receives the requested tokens can then use them to access the data it wants from the File Exporter.

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

Tokens

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

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

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 wishes to manipulate; for instance, appending text to a file requires open tokens to access the file, data tokens to modify the contents of the file, and status tokens to change the size of the file.

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

Token Management

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

1. 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. (Information about token revocation follows this list.)
2. 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:

1. 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.
2. If the client's host lifetime has expired but its host RPC lifetime has not, the File Exporter tries to contact the client one time. If the client responds, the File Exporter cannot revoke the client's tokens without its permission; otherwise, the File Exporter can revoke any tokens held by the client without contacting it further.
3. If the client's host RPC lifetime has expired, the File Exporter can revoke the client's tokens without contacting it.

Token State Recovery

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

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

During the first form of token state recovery, the File Exporter attempts to preserve the state of its tokens across restarts by initially accepting requests only to re-establish 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 re-establish 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 re-establish 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 process. (See “fxd” on page 722 for complete information about the **fxd** process and its options.)

Data Communication Security in DFS

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

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

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

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

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

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

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

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

You can establish the upper and lower File Exporter RPC authentication bounds by using the **fxd** command. You cannot display a File Exporter's RPC authentication bound settings. For more information about setting the File Exporter's authentication bounds with the **fxd** command, see “fxd” on page 722.

DFS Distributed Database Technology

DFS includes two administrative databases:

- Fileset Location Database (FLDB)
- Backup Database.

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

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

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

Ubik Database Synchronization

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

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

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

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

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

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

Because Ubik relies on the actions of a quorum, having an odd number of database sites is helpful; in most cases, storing a replicated database at three sites is sufficient. However, 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 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:

- It stops receiving votes from a quorum of the database sites.
- It cannot propagate changes to a quorum of the database sites.
- It 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, with the vote of the site with the lowest network address of all database server machines of that type again carrying 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. During the election and distribution, the database is unavailable. The election and database distribution are typically brief, usually taking no longer than a few minutes.

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

Providing Information for Ubik

For the most part, Ubik operates without human intervention. However, it does depend on other DCE facilities and services for some things. The following list describes the interaction between Ubik and the remainder of DCE. It also provides an overview of the configuration information necessary for Ubik to operate properly. "Configuring Database Server Machines for Ubik" on page 71 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 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 allows 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**.

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

Configuring Database Server Machines for Ubik

A cell's initial database server machines are configured when DFS is installed and configured in the cell. If it becomes necessary to add or remove database server machines after initial cell configuration, perform the steps in “Adding a Database Server Machine” on page 72 and “Removing a Database Server Machine” on page 73 to properly configure information for Ubik. You may be able to use the DCE installation and configuration script to modify the database servers configured in your cell. See the *OS/390 DCE Administration Guide* for more information about the script.

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, at which time 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 through that machine. Eventually, all machines will 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.

The following sections 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 “udebug” on page 761 for complete information about the **udebug** command and its options.) See 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 9, “Monitoring and Controlling Server Processes” on page 129 for more information about **bos** commands.)

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 10, “Making Filesets and Aggregates Available” on page 147 for more information about using the **fts crserverentry** command to create server entries.)
2. Use the **dcecp group add -a** 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**). This 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 through 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. Ensure the appropriate administrative list (**admin.fl** or **admin.bak**) has been propagated to the new database server machine. These administrative lists are typically propagated from the cell's System Control machine by the Update Server. Modify the Update Server as necessary if the list is propagated from the cell's System Control machine.

   The following example verifies that the **admin.bak** administrative list has been propagated to the new database server machine, **DCEDFS**, by issuing the following:

   ```
   $ bos lsa /.:/hosts/DCEDFS -adminlist admin.bak

   AdminUsers are: user: jones, user: smith,
   user: hosts/DCEDFS/dfs-server, group: dfs-admin, group: fs1-admin
   ```

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 the processes to read the updated RPC server group. This ensures that each Ubik coordinator agrees on the number and identities of the other database server machines of its type, which 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.
8. Using the **udebug** command verifies that all **flservers** or all **bakservers** in the cell are all synchronized with each other in order for distribution of changes to occur at all secondary sites. If at least two of the **flserver** or **bakserver** machine times are more than 40 seconds apart, distribution does not occur and the message `****clock may be bad` will be in the output from the **udebug** command.

   If that is the case, verify that you are running a **dtsd** server on OS/390. If it is running on a different subnet than any other **dtsd** local server in the cell, make sure that it is configured as a global server.

   If that is not the case, modify the time on any other machines housing additional **dtsd** local or global servers, or modify the time of DCEKERN using the **modify DCEKERN, clock set** command so that you get all machines housing the **flservers** and **bkservers** within 40 seconds of each other.

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 in the group domain 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 the processes to read the updated RPC server group. This ensures that each Ubik coordinator agrees on the

number and identities of the other database server machines of its type, which 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 by 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 **/opt/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.

Large File Support in DFS

OS/390 DFS supports a maximum file size of greater than 2 Gigabytes (or $2^{31}-1$ bytes). To use files greater than 2GB, both the DFS client and server have to support large files. DFS clients and servers that do not support large files encounter errors attempting to access large files. The actual error returned to the application is system dependent.

To ensure that the DFS file operations work properly when large files are present in the DFS filespace, the system should be configured to allow large files (for example, see the MAXFILESIZE parameter of the BPXPRMxx parmlib member in the *OS/390 UNIX System Services Planning* book, SC28-1890).

Chapter 5. Using ACLs and Groups

Important Note to Users

This chapter and other chapters in this book discuss the **dcecp acl** command. It is important to note that you cannot use the **dcecp acl** command on exported non-Local File System files. In OS/390 DFS, that means you cannot use the **dcecp acl** command on exported OS/390 HFS or RFS files.

This section 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 Local File System filesets. (See the DCE Security Service portion of the *OS/390 DCE Administration Guide* for details about manipulating DCE ACLs.)

This section also presents information about groups. In addition to using groups in ACLs, you can use them 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 6, “Using Administrative Lists and Keytab Files” on page 103 for complete details about using administrative lists; see the *OS/390 DCE Administration Guide* for information about creating and maintaining groups.)

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

Using DCE ACLs with DFS

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

All file and directory objects in DCE Local File System 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 to your directories and four different permissions to 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 Local File System filesets; they do not replace them. DCE Local File System ensures that an object's mode bits and its ACL permissions are always synchronized. Note that objects in DCE Local File System filesets can rely exclusively on mode bits as their sole form of protection. (See “Initial Protection of a New File or Directory” on page 86 and “Initial ACLs of a New Fileset” on page 96 for more information about this possibility; see “ACL Interaction with UNIX Mode Bits” on page 85 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 Local File System filesets. Mode bits are the only form of protection used with objects in most non-Local File System filesets.

ACL Entries

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

```
{type [key] permissions}
```

The elements of an entry provide the following information:

- The *type* specifies the kind of user or group to which the entry applies.
- The *key* names the specific user or group to which the entry applies. Some entries apply to predefined collections of users and groups 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 Local File System objects can include six access permissions: READ (**r**), WRITE (**w**), EXECUTE (**x**), CONTROL (**c**), INSERT (**i**), and DELETE (**d**).

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: The text of this chapter refers exclusively to ACL entries for users and groups. However, an ACL entry can apply to any principal (for example, to a server principal).

ACL Entry Types for Users and Groups

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

An ACL's default cell names the cell with respect to which the ACL's entries are defined. A user or group named in an ACL entry is assumed to be from the default cell unless the entry explicitly names a different cell. The default cell is not necessarily the cell in which the ACL exists. For example, an object in cell **abc.com** can have an ACL whose default cell is **def.com**. With respect to ACLs, a local user is one whose local cell is the same as the default cell of an ACL; conversely, a foreign user is one whose default cell is different from the default cell of an ACL. Table 6 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 6 (Page 1 of 2). 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*	A specific user (*username*) from the default cell.
foreign_user	*cell_name/username*	A specific user (*username*) from a specific 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 a specific group (*group_name*) from the default cell.

Table 6 (Page 2 of 2). ACL Entry Types for Users and Groups

Type	Key	Applies to
foreign_group	*cell_name/group_name*	Members of a specific group (*group_name*) from a specific 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 a specific foreign cell (*cell_name*) who do not match any of the preceding entries.
any_other	None	Users from any foreign cell who do not match any of the preceding entries.

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

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

For instance, a **user** entry specifies the permissions for a user whose local cell is the same as the default cell of an ACL. Whereas the entry types in the previous list refer to users and groups whose local cell is the same as an ACL's default cell, the **foreign_user**, **foreign_group**, **foreign_other**, and **any_other** entry types refer to users and groups whose local cells are different from an ACL's default cell. For instance, a **foreign_user** entry specifies the permissions for a user whose local cell is different from the default cell of an ACL. (Note that **foreign_** entries can exist for users or groups from the default cell. See "The Default Cell and ACL Inheritance" on page 87 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 a user named **frost** from the local cell.

{group writers *permissions***}** Defines the permissions for a group named **writers** from the local cell.

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

{foreign_group /.../abc.com/writers *permissions***}**
Defines the permissions for a 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 Local File System 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 the same key) can exist on an ACL. For example, only one **user** entry can exist for a given *username* from the local cell.

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

ACL Entry Types for Masks

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

{**mask_obj** *permissions*}

The **mask_obj** entry specifies the maximum set of permissions that can be granted by any entries *except* the **user_obj** and **other_obj** entry types. 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 Local File System 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 Local File System 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 Local File System 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 Local File System objects, not ACLs on objects associated with other DCE components. DCE Local File System enforces this restriction in an effort to track Draft 12 of the POSIX standard for ACLs on file and directory objects.

ACL Entry Types for Unauthenticated Users

An unauthenticated user is one whose DCE identity has not been verified by the DCE Security Service. For example, a user can access the DCE without being authenticated by logging into the local machine without logging into the DCE. In this case, the user is said to be unauthenticated because the 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 a DCE Local File System object, the DFS first determines whether the user is authenticated. An authenticated user acquires the permissions associated with the user's authenticated identity according to the normal ACL evaluation routine. An unauthenticated user's permissions are determined as follows:

1. The DFS identifies the user as **nobody**, regardless of the cell from which an unauthorized user requests access to an object. The DFS 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 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 user ID and group ID of the identity **nobody** are both typically **-2**, but they can vary between File Server machines.

2. DCE Local File System 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 an object in a non-Local File System fileset, 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 Local File System filesets.

Note: The DCE ACLs used with objects for other DCE components include an additional **unauthenticated** entry type that serves as a mask for unauthenticated users. Prior to DCE 1.1 (note, OS/390 Release 2 provides DFS at the DCE 1.1 level), DCE Local File System allowed **unauthenticated** entries to be included on the ACLs of DCE Local File System objects, but it ignored the entries when determining users' permissions.

As of DCE 1.1, DCE Local File System no longer allows **unauthenticated** entries to be included on the ACLs of DCE Local File System objects. It is possible for the ACLs of existing objects to include **unauthenticated** entries added with earlier versions of DCE Local File System. DCE Local File System 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 Local File System objects. Moving or restoring a DCE Local File System fileset to a File Server machine that is running DCE 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 Local File System objects in your cell.

ACL Permissions

Each ACL entry for a user or group includes a set of permissions to define 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 Local File System 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.

- READ (**r**)
- WRITE (**w**)
- EXECUTE (**x**)
- CONTROL (**c**)
- INSERT (**i**)
- DELETE (**d**).

Table 7 on page 80 lists the various operations that can be performed on a file or directory and the ACL permissions that are required to perform them. As the table indicates, all operations performed on a file or directory object require the EXECUTE (**x**) 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 sections; not all operations explicitly list it.

Note: A user must have the EXECUTE (**x**) 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 7. 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
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 7, 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 **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 to have 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 be deleted.

For example, suppose the user **rajesh** needs to execute the DFS **fms** command to determine the tape capacity and file mark size of a tape drive to be used with the DFS Backup System. 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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) permissions on the directory from which the command is issued, as well as the EXECUTE (**x**) permission on each directory that leads to the directory.

The following example ACL entry grants **rajesh** the **w**, **x**, and **i** permissions necessary 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**, **c**, and **d** permissions.

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

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

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

ACL Evaluation

When a user tries to perform an operation on an object, DCE Local File System 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 Local File System evaluates the entries on the object's ACL in a specific order. The following evaluation sequence defines the order in which DCE Local File System checks ACL entries. ACL evaluation stops once the user matches the conditions in one of the steps in the evaluation sequence. Therefore, it proceeds to the next step in the sequence only if the user failed to match the conditions in all of the previous steps. (See Table 6 on page 76 for a description of the different ACL entry types referred to in the following list.)

1. The user matches the **user_obj** entry. If the **user_obj** entry applies, DCE Local File System grants the user the permissions specified in the 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. The user matches a **user** or **foreign_user** entry. If a **user** or **foreign_user** entry on the ACL applies, DCE Local File System grants the user the specified permissions after filtering them through the **mask_obj** entry.
3. The user matches one or more **group_obj**, **group**, or **foreign_group** entries. If one or more group-related entries on the ACL apply, DCE Local File System grants the user all of the permissions accrued from the applicable group entries after filtering them 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 Local File System grants the user the permissions specified with the **other_obj** entry. The permissions are not filtered through the **mask_obj** entry.
5. The user matches a **foreign_other** entry. If a **foreign_other** entry on the ACL applies, the DCE Local File System grants the user the specified permissions after filtering them through the **mask_obj** entry.
6. The user is from a foreign cell that does not have a **foreign_other** entry. DCE Local File System 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. If no entry applies, DCE Local File System denies the user access.

Before DCE Local File System 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 steps. If the user is not authenticated, the 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 Local File System 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 through 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 Local File System 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 Local File System denies **dale** WRITE access for the file.

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

ACL Evaluation for Local Access

On some operating systems, such as AIX, the **mount** command allows a DCE Local File System fileset to be mounted locally, as a file system on its File Server machine. You can access an object in a locally mounted DCE Local File System fileset through a local path, as well as through a DCE path. The same ACL evaluation algorithm is used to determine your permissions in either case. However, if your local identity is different from your DCE identity, the permissions you receive when you access the object through its local path are those associated with your local identity, while the permissions you receive for access through the object's DCE path are those associated with your DCE identity. This is also true of objects in non-Local File System filesets.

Important Note to Users

On OS/390 DFS, you cannot mount a DCE Local File System fileset locally in your HFS hierarchy.

For example, suppose you log into the local machine as **root** and then authenticate to the DCE as your DCE identity. In this case, if you access an object through its local path, you receive **root** permissions for the object; if you access the same object through its DCE path, 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 through its local pathname, you receive **root** permissions; however, if you access the object through 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 through its DCE pathname, you can include an entry for the machine's **self** identity on the ACL of the object. The **self** identity can also receive permissions from a group to which it belongs or from the **other_obj** entry (because it is treated as an authenticated user from the local cell).

Setting and Examining ACLs

The **dcecp acl** command is used to list and modify the ACLs of DCE Local File System objects. In most respects, the operation of the **dcecp acl** command with DCE Local File System 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 EXECUTE (**x**) 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 CONTROL (**c**) permission for the object, as well as the EXECUTE (**x**) 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 Local File System 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 *OS/390 DCE Administration Development Guide Core Components* for complete details about using the command to set and examine an object's ACLs.)

Rules for Modifying ACLs

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

- The **user_obj**, **group_obj**, and **other_obj** entries must always exist. All other entry types are always 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 already 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 already exists on the ACL.)

Examples of Listing and Modifying an ACL

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

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

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

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

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

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

The following example performs both of these 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 this 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 Local File System 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 through the **user** entry.

ACL Interaction with UNIX Mode Bits

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

Each type of user (**user**, **group**, and **other**) can be assigned any combination of the **r**, **w**, and **x** permissions through 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 Local File System filesets, mode bits are the only form of protection associated with objects in most non-Local File System filesets. In DCE Local File System filesets, all file and directory objects can have both UNIX mode bits and DCE ACLs. Note that all objects always have UNIX mode bits, but they do not necessarily have ACLs. (See “Initial Protection of a New File or Directory” on page 86 for more details.)

For DCE Local File System objects, DCE Local File System 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 Local File System 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 appropriate ACL **_obj** entry (**user_obj**, **mask_obj**, **group_obj**, or **other_obj**), DCE Local File System 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 Local File System 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 Local File System 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, when you use the UNIX **chmod** command to modify the mode bits associated with an object, DCE Local File System reconciles the corresponding ACL entries. Thus, DCE Local File System 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, this means that at least one of the following ACL entries must have the **x** permission for the file to be executed: **user_obj**, **mask_obj** (or **group_obj**, if the **mask_obj** entry does not exist on the ACL), or **other_obj**.

Initial Protection of a New File or Directory

Each DCE Local File System file or directory can have an Object ACL that controls access to the file or directory; all previous examples in this section 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 object 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, initially used 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 Local File System ensures that the object's mode bits and its ACL permissions are always synchronized.
- If a new object's parent directory does *not* have the appropriate Initial Creation ACL, the new object initially has no Object ACL; the object relies on mode bits as its only form of protection. If they are not inherited, ACLs can be explicitly created with the **dcecp acl** command. (See “Mode Bits for New Objects That Do Not Inherit ACLs” on page 94 for information about using the **dcecp acl** command to create a directory's initial ACLs.)

  Note: An Object ACL is always created for a file or directory that is created by a foreign user, even if the parent directory does not have the appropriate Initial Creation ACL. (See “ACL Inheritance for Objects Created by Foreign Users” on page 91 and “Mode Bits for New Objects That Do Not Inherit ACLs” on page 94 for more information.)

The following sections describe how the initial protections of a new object are derived. The first section discusses 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; note that both of these subsections assume that the parent directory has the proper Initial Creation ACL. The final subsection discusses how the UNIX mode bits are determined for an object whose parent directory does not have the appropriate Initial Creation ACL.

The Default Cell and ACL Inheritance

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

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

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

  For example, an entry of the type **user** *username* specifies the permissions for the user *username* whose local cell is the same as the default cell of the ACL.
- A foreign user is one whose local cell is different from the default cell of an ACL. The following entry types are defined for foreign users and groups:
  - **foreign_user**
  - **foreign_group**
  - **foreign_other**
  - **any_other**.

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

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

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

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

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

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

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

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

Although you can make the default cells of a directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL different from one another, this 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: Changing an ACL's default cell by including the **-cell** option with the **dcecp acl modify** command changes the meaning of the ACL's entries. For example, the **other_obj** entry no longer applies to users from the former default cell; it now applies to users from the new default cell. Also, entry types such as **user_obj** and **user**, which are defined with respect to an ACL's default cell, can now give permissions to different users in the new default cell.

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

ACL Inheritance for Objects Created by Local Users

For ACLs, a local user is a user whose 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, the DCE Local File System uses the intersection of the following information to determine the Object ACL that it creates for the 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 resulting system call typically specifies the **user**, **group**, and **other** mode bits as **r** and **w**.

- 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.
- The UNIX **umask** of the process (for example, the login process of a user) that creates the object. (The UNIX **umask** filters the mode bits initially assigned to an object; it is defined as the octal complement of the allowable mode bits.)

For example, when a file is created, the DCE Local File System 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, masked by the **umask** of the process that creates it. 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, masked by the **umask** of the process that creates it.
- 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, masked by the **umask** of the process that creates it. 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, masked by the **umask** of the process that creates it. 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.

The DCE Local File System 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 7 on page 90 illustrates ACL inheritance for files and directories.

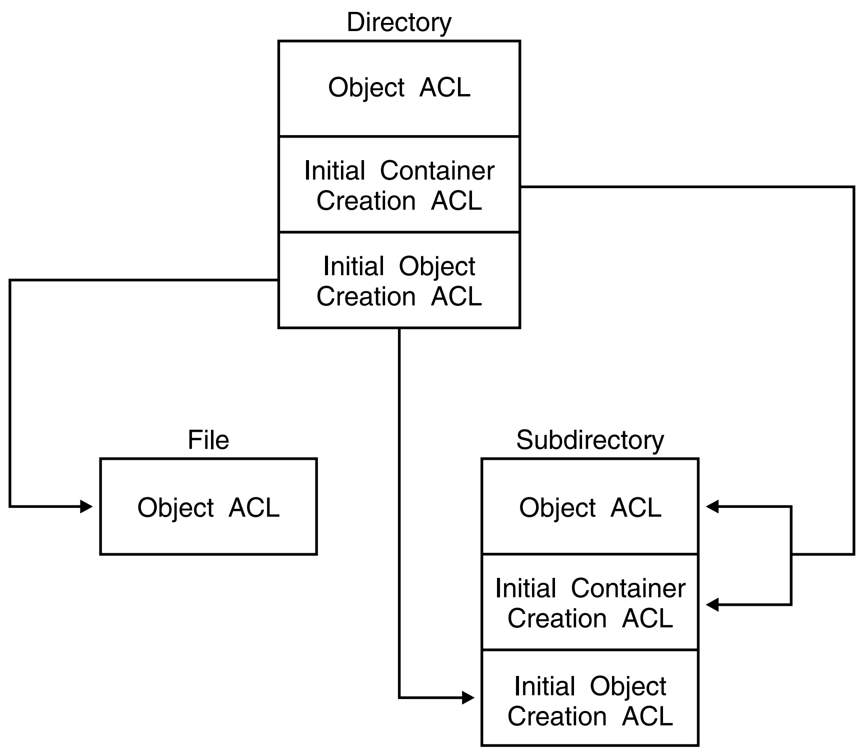

Directory
Object ACL
Initial Container Creation ACL
Initial Object Creation ACL
File
Object ACL
Subdirectory
Object ACL
Initial Container Creation ACL
Initial Object Creation ACL

Figure 7. 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 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 pierette** entry 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).

ACL Inheritance for Objects Created by Foreign Users

Any user who has the **w**, **x**, and **i** permissions on a directory can create objects in the directory. This is true regardless of whether the user's local cell is different from the default cell of the directory (that is, regardless of whether the user is a foreign user with respect to the directory). 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 user creates an object, ACL inheritance occurs as described in “ACL Inheritance for Objects Created by Local Users” on page 88. 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 user who is foreign with respect to the default cell of a directory's Object ACL creates an object in the 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.

The permissions granted by the Object ACL are based on the UNIX mode bits specified at the system call level when the object is created and on the value of the UNIX **UMASK** variable of the creating process. (See “Using the -noauth Option” on page 109 for more information.) Note that 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. (See “ACL Inheritance for Objects Created by Foreign Users” on page 91 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**, the user **srivas**'s local cell. For example, the **user pierette** and **user vijay** entries apply to specific users from the cell **abc.com**, and the **other_obj** entries apply to other 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 entries 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 andi** entry 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 the cell who are not granted permissions from specific user or group entries are now granted permissions from the **other_obj** entry. The **foreign_other /.../def.com** entry 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 cells of the Initial Object Creation ACL and Initial Container Creation ACL of the parent directory of the **andi_files** subdirectory, entries on the corresponding ACLs that the subdirectory inherits are also changed as necessary. The new subdirectory inherits the following Initial Object Creation ACL and Initial Container Creation ACL from its parent:

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

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

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

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

Mode Bits for New Objects That Do Not Inherit ACLs

The ACL inheritance described in “ACL Inheritance for Objects Created by Local Users” on page 88 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.

When a file or directory object is created in a directory that does not have the proper Initial Creation ACL, the DCE Local File System uses the intersection of the following information to determine the object's initial mode bits:

- The UNIX mode bits specified at the system call level (with the UNIX **open()**, **creat()**, or **mkdir()** system call) when the object is created. For example, when the UNIX **touch** command is used to create an object, the resulting system call usually specifies the **user**, **group**, and **other** mode bits as **r** and **w**.
- The value of the UNIX **UMASK** variable of the process that creates the object. The **UMASK** variable filters the mode bits initially assigned to an object; the variable is defined as the octal complement of the allowable mode bits. For example, if a user creates an object, the value of the **UMASK** variable of the user's login 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 grants READ and WRITE access to **user**, **group**, and **other**; the **UMASK** variable 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 grants **r**, **w**, and **x** access to **user**, **group**, and **other**; the **UMASK** variable commonly restricts **group** and **other** to only **r** and **x** 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. Because its parent directory has no Initial Container Creation ACL, the directory is also created without an Initial Container Creation ACL. If the parent directory has an Initial Object Creation ACL, the new directory inherits it; otherwise, the new directory is created without an Initial Object Creation ACL.

Displaying Implicit (Nonexistent) ACLs: If you use **dcecp acl show** command to display a nonexistent Object ACL, DCE Local File System displays an implicit Object ACL. Although an Object ACL does not physically exist, DCE Local File System 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 Local File System 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 Local File System 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 Local File System maps the **w** mode bit to just the **w** ACL permission, not the **i** and **d** permissions (see “ACL Interaction with UNIX Mode Bits” on page 85). Because the **i** and **d** ACL permissions are meaningless for a file, DCE Local File System 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 Local File System displays an implicit Initial Creation ACL. It bases the permissions of the implicit Initial Creation ACL solely on a file creation mask of **0** (zero). In this case, DCE Local File System 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 Local File System 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 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 Local File System directory that does not have the appropriate Initial Creation ACL, DCE Local File System 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 Local File System ignores the file creation mask of the calling process only when displaying nonexistent Initial Creation ACLs.

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

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

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

Initial ACLs of a New Fileset

The root directory of a newly created DCE Local File System fileset has null ACLs. Until the **dcecp acl** command is used to create the Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL for the root directory, the directory has no ACLs; it is protected only with UNIX mode bits. Files and subdirectories created in the directory are assigned mode bits according to the usual file system semantics described in the previous section.

A DCE Local File System 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 Local File System fileset, **user**, **group**, and **other** all receive the UNIX **r**, **w**, and **x** permissions. If the **dcecp acl show** command is used to view the directory's Object ACL, DCE Local File System 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 Local File System displays an implicit Initial Creation ACL. DCE Local File System constructs implicit Initial Creation ACLs for a new root directory just as it constructs implicit Initial Creation ACLs for any directory that does not have these ACLs. (See “Displaying Implicit (Nonexistent) ACLs” on page 95.)

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 Local File System 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 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.

Using Groups with DFS

Information about groups is maintained in the Registry Database by the DCE Security Service. (See the *OS/390 DCE Administration Guide* 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 section does not discuss the management of administrative lists in detail. (See Chapter 6, “Using Administrative Lists and Keytab Files” on page 103 for details about creating and maintaining administrative lists.)

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

Creating and Maintaining Groups

To authenticate to the DCE, users must have accounts in the Registry Database (although some parts of the 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.

Note: 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 *OS/390 DCE Administration Guide* and *OS/390 DCE Command Reference* for information about the **dcecp group** command.)

Using Groups with ACLs, Administrative Lists, and Commands

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

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

Similarly, administrative lists specify the users and groups that can perform actions affecting specific server processes. Groups can be specified on the administrative list associated with each DFS server process. Often the same users need to be included on several administrative lists; these users can be specified as a group in the Registry Database and subsequently added to, and removed from, administrative lists as a group. For example, you can use a group to specify a system administration team whose members need access to most DFS servers. Then, rather than modify all the administrative lists when the team membership changes, you can simply 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** process and the **fts crserverentry** command, 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, while multiple groups can be specified with ACLs and administrative lists.

Suggestions for Administrative Groups

Administrative lists determine who is 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 6, “Using Administrative Lists and Keytab Files” on page 103 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 **fxd** process 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. They have the equivalent of the ACL **c** permission on all of the files and directories in each exported DCE Local File System fileset, and they can effectively change the UNIX permissions on all of the files and directories in each exported non-Local File System 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 Local File System object exported from the machine. Because they have access to all of the exported DCE Local File System and non-Local File System filesets on the File Server machine, members of this group should be both few in number and highly trusted.

While 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 through its DCE pathname or through its local pathname.

The first way a user can access a file or directory is through 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 the DCE as **root**, the DFS treats the user according to **root**'s DCE identity. The DCE identity **root** effectively has **root** privileges for data in all exported non-Local File System filesets in the cell, which is a serious security risk. Use the DCE **root** identity very cautiously or disable it altogether.

Important Note to Users

In OS/390 systems, remote clients with the DCE identity **root**, do not necessarily have **root** privileges on OS/390 exported non-Local File System filesets. See “Mapping DCE User IDs to OS/390 User IDs” on page 174 for information on user ID mapping.

The second way a user can access a file or directory is through 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-Local File System fileset can always be accessed through a local path because a non-Local File System fileset must always be mounted locally, as a file system on its File Server machine; a file or directory in a DCE Local File System fileset can be accessed through a local path 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 first changing the protection. Being authenticated as DCE **root** lets you perform any operation on a file or directory in an exported non-Local File System fileset immediately.

Important Note to Users

In OS/390 systems, remote clients with the DCE identity **root**, do not necessarily have **root** privileges on OS/390 exported non-Local File System filesets. See “Mapping DCE User IDs to OS/390 User IDs” on page 174 for information on user ID mapping.

The **fts crserverentry** command 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 of the filesets in the domain. Specifying a group with **fts crserverentry** 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**), and
- 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 8 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 8. Suggested Groups for Administering a Single-Domain Cell

Administrative List	With Two Groups	With Three Groups	Function
admin.bos	**cell_fileset**	**cell_file_system**	Manage server processes on each server machine
admin.fl	**cell_fileset**	**cell_fileset**	Create server and fileset entries in the Fileset Location Database on each Fileset Database machine
admin.ft	**cell_fileset** **cell_servers**	**cell_fileset** **cell_servers**	Manage filesets on each File Server machine; move filesets between File Server machines
admin.bak	**cell_fileset**	**cell_fileset**	Modify the Backup Database on each Backup Database machine
admin.up	**cell_servers**	**cell_servers**	Allow **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** process 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** process 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** process 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 10, “Making Filesets and Aggregates Available” on page 147 for more information about creating server entries.)

Chapter 6. Using Administrative Lists and Keytab Files

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.)

Important Note to Users

The following items noted in this chapter are not available in OS/390 DFS:

- OS/390 cannot act in a binary distribution role on either client or server.
- The Private File Server Machine.

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 5, “Using ACLs and Groups” on page 75 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.

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 Chapter 19, “Distributed File Service Commands” on page 401 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 File Server machine using the full DCE pathname (for example, **/.../abc.com/hosts/fs1**).

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

  – The machine's DCE pathname (for example, **/.../abc.com/hosts/fs1**) and the **-noauth** option
  – The machine's hostname (for example, **fs1.abc.com** or **fs1**)

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

Note: If you specify the hostname 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 hostname 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 **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled (with the **bos setauth** command), the identity **nobody** has the necessary privileges to perform any operation. (See "Disabling DFS Authorization Checking on a Server Machine" on page 108 for more information.) If you use this option, do not use the **-localauth** option.
- The **-localauth** option directs **bos** to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server** (for example, **/.../abc.com/hosts/fs1/dfs-server**). Do not confuse a machine's DFS server principal with its unique **self** identity. (See Chapter 10, "Making Filesets and Aggregates Available" on page 147 for information about DFS server principals.)

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

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

Using Administrative Lists

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

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

To simplify the management of these lists, use the domain's System Control machine as the source of all administrative lists for the domain. That 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 machines in the domain share the administrative lists and, therefore, share a common set of administrators. (See "upserver" on page 768 and "upclient" on page 765 for more information).

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 housing 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. (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 documentation, the specific privileges required to execute commands are detailed with each task. (See Chapter 19, "Distributed File Service Commands" on page 401 for complete information about the administrative privileges and permissions required to issue each DFS command.) Note that the names of the administrative lists are only recommendations; different names can be specified when the respective processes are started.

Maintaining Administrative Lists

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

- A server process automatically creates its administrative list when it is started on a machine if the list does not already exist on the local disk of the machine. By default, a process places its list in the configuration directory (**/opt/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 (**/opt/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 **/opt/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.

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

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

Listing Principals and Groups in Administrative Lists:

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

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

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

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

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

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

Adding Principals and Groups to Administrative Lists: To add principals and groups to an administrative list, perform the following steps:

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 (**/opt/dcelocal/var/dfs**). If the lists are stored in the default directory, you need to provide only the specific file name (**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 (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 added to the list. A group from the local cell can be specified by a full or abbreviated group name (for example, **/.../***cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.
   - The **-createlist** option specifies that the administrative list indicated with **-adminlist** is to be created if it does not already exist. Any principals or groups specified with the command are added to the new file; if no principals or groups are specified, the command creates an empty file. This option has no effect if the specified file already exists.

Removing Principals and Groups from Administrative Lists: To rename principals and groups for an administrative list, perform the following steps:

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 (**/opt/dcelocal/var/dfs**). If the lists are stored in the default directory, you need to provide only the specific file name (**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 (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 **-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.

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; and
- Manually creating the zero-length file **/opt/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, disable checking 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, disable checking by manually creating the **NoAuth** file. If the **secd** process or a related security process is unavailable, the issuer of a command cannot acquire the security credentials necessary to allow DFS server processes to verify administrative privilege. In this case, the **-noauth** option must be included with a command to bypass the unavailable Security Service. (See “Using the -noauth Option” on page 109.)
- During server encryption key emergencies, disable checking 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, disable checking by issuing the **bos setauth** command. If authorization checking is enabled, checksums are displayed rather than the actual keys. (See “Using Keytab Files” on page 110.)

Never disable DFS authorization checking for longer than is absolutely necessary. Disabling DFS authorization checking on a machine compromises security by allowing anyone, including the unprivileged identity **nobody**, to execute any DFS command on the machine. To enable DFS authorization checking (the normal state) once it has been disabled, use the **bos setauth** command. Use the **bos status** command to determine whether DFS authorization checking is enabled or disabled on a server machine.

Using the -noauth Option: Most DFS commands from the **bos** and **fts** suites have an optional **-noauth** option. Omitting the **-noauth** option from a command requires that authentication information be available about the issuer of the command; since the option is not required, it 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 the following conditions apply:

- **Authentication information is unnecessary.** If DFS authorization checking is disabled on a server machine or if a command does not require administrative privilege, DFS server processes on the machine do not check for authentication information. Omitting the **-noauth** option in these cases causes the **bos** or **fts** program to include the unnecessary security credentials of the issuer with the command; including the **-noauth** option allows the command to execute more quickly because it avoids the unnecessary creation of the issuer's security credentials.

  You may want to include the **-noauth** option with a command that does not require administrative privilege if the command is to be issued as a **bos cron** process. (See Chapter 9, “Monitoring and Controlling Server Processes” on page 129 for more information about **bos** processes.)

- **Authentication information is unavailable.** If some aspect of the Security Service is unavailable (for example, if the **secd** process is not functioning) and the **-noauth** option is omitted from a command, **bos** and **fts** commands fail even if DFS authorization checking is disabled. The failure occurs because the **bos** or **fts** program cannot obtain the issuer's security credentials from the Security Service. In such cases, even commands that do not require administrative privilege may fail if the **-noauth** option is not used.

Including the **-noauth** option when DFS authorization checking is disabled ensures that a command will succeed because the Security Service is never contacted to assemble the issuer's security credentials. Include the **-noauth** option with a command that requires administrative privilege only if DFS authorization checking is disabled on the necessary machines. A command that requires administrative privilege fails if the **-noauth** option is included and DFS authorization checking is not disabled.

Disabling or Enabling DFS Authorization Checking: To disable or enable DFS authorization checking, perform the following steps:

Attention: Disabling DFS authorization checking makes potentially serious security breaches possible. Enable DFS authorization checking as soon as the need to have it disabled has passed.

1. If DFS authorization checking is to be disabled, verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which checking is to be disabled. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list.
2. Enter the **bos setauth** command to disable or enable DFS authorization checking on the server machine:

   ```
   $ bos setauth -server machine -authchecking {on | off}
   ```

   The **-authchecking off** command disables DFS authorization checking by creating the **NoAuth** file on the machine specified with **-server**; the **-authchecking on** option enables DFS authorization checking by removing the **NoAuth** file from the machine specified with **-server**.

Using Keytab Files

An encryption key is a set of octal characters used to encrypt and decrypt packets of information. In DFS, a server encryption key is employed to provide security for information transferred between server processes and their clients. An encryption key for a server is analogous to a password for a user. All DFS server processes on a server machine use the same key from the keytab file as a password for that machine.

One or more keys are stored in the **/krb5/v5srvtab** keytab file on the local disk of each server machine. Each key is associated with a principal name, usually the DFS principal name of the machine on which the key resides. Multiple keys can be associated with a principal name in a keytab file, but one key (usually the most recent) is also stored in the Registry Database for any principal name in a keytab file.

The key stored in the Registry Database is the one used for subsequent communications between processes on client machines and processes on the server machine. Multiple keys can exist if a new key is added while an existing key is still being used for communications between a client and server. Note that once communications have been initiated between a client and server using a key, removing that key may not prevent any further communications between the two.

Maintaining Keytab Files

Maintaining server encryption keys and keytab files is critical to establishing adequate security measures in your cell or domain. Under normal circumstances, keytab files require little maintenance. Because they are analogous to user passwords, they should be changed about as often.

The first step in changing a server encryption key is to add a new key to the keytab file. Two commands are available for adding keys — **bos genkey** and **bos addkey**.

- The **bos genkey** command automatically generates a random key. It also automatically updates the entry in the Registry Database for the principal with which the key is associated. Any subsequent communications that involve the specified principal and that require a key use the newly added key.
- The **bos addkey** command performs a similar function, but it requires that you enter a string to be converted into a key, and it gives you the option of updating the Registry Database entry for the indicated principal. The **bos addkey** command is less secure than the **bos genkey** command because user-specified strings are seldom as random as machine-generated strings.

A keytab file must already exist before either of these commands can be used to add a key to it. (Keytab files are created with the **dcecp keytab create** command.)

A unique version number is associated with each key for a principal in a keytab file. When adding a key to a keytab file, you must specify its key version number in one of the following formats.

- 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. (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 two 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 the DCE. If it is not available to you, the BOS Server uses the next-highest protection level, packet integrity. It displays the following message, reporting that it must use the packet integrity protection level because packet privacy is not available:

```
Data encryption unsupported by RPC. Continuing without it.
```

Listing Keys in Keytab Files:

To list the keys in a keytab file, perform the following steps:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine whose keys are to be displayed. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Issue the **bos lskeys** command to view the key version numbers and checksums from the keytab file on a server machine:

   ```
   $ bos lskeys -server machine [-principal name]
   ```

   The **-principal** *name* option is the principal name for which associated keys are to be listed. The default is the DFS principal name of the machine specified with **-server**.

Adding Keys to Keytab Files:

To add a key to a keytab file, perform the following steps:

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. 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 *OS/390 DCE Administration Guide* 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. The following are valid arguments for this option.
     - 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. The following are valid arguments for this option.
     - 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 using the **bos addkey** command and its **-localonly** option, use the **dcecp keytab add** command with the **-registry** option to add the key to the Registry Database when necessary.

Removing Specific Keys from Keytab Files: To remove a specific key from a keytab file, perform the following steps:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the keytab file to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Remove one or more keys from the keytab file with the **bos rmkey** command:

   ```
   $ bos rmkey -server machine -kvno version_number... [-principal name]
   ```

   - The **-kvno** *version_number* option is the key version number of each key to be removed for the indicated principal. Valid arguments for this option are integers in the range 1 to 255.
   - The **-principal** *name* option is the principal name associated with the keys to be removed from the keytab file. The default is the DFS principal name of the machine specified with **-server**.

Removing All Obsolete Keys from Keytab Files: To remove all obsolete keys from a keytab file, perform the following steps:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the keytab file to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Remove obsolete keys (those keys that are no longer in use) from the keytab file with the **bos gckeys** command:

   ```
   $ bos gckeys -server machine [-principal name]
   ```

   The **-principal** *name* option is the principal name for which obsolete keys are to be removed from the keytab file. The default is the DFS principal name of the machine specified with **-server**.

Handling Server Encryption Key Emergencies

Server encryption key emergencies are situations that require immediate attention to ensure continued, authenticated communications between the processes on a server machine and the clients with which they are communicating. One type of emergency occurs when you suspect that a machine's encryption key in the Registry Database is compromised. In this case, you must immediately remove that key from the keytab file and reboot the server machine to prevent unwanted access to the server.

Important Note to Users

In OS/390 DFS, stopping and starting the DFS started task is the equivalent of rebooting a server machine. See Chapter 7, “Starting and Stopping the DFS Server in OS/390 DFS” on page 117 for instructions on stopping and starting DFS on OS/390.

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. (See Chapter 9, "Monitoring and Controlling Server Processes" on page 129 for a description of the **bos shutdown** command.)

Creating New Encryption Keys: To create a new encryption key, perform the following steps:

1. Log in as **root** on the affected machine.
2. Disable DFS authorization checking by creating the **/opt/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 **bos** 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. The following are valid arguments for this option.
     – 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 **bos** 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 **bos** 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 **bos** to use the unprivileged identity **nobody** as the identity of the issuer.

7. Enable DFS authorization checking by entering the **bos setauth** command. Specify the value **on** with the **-authchecking** option, and include the **-noauth** option.

```
# bos setauth -server machine -authchecking {on | off} —noauth
```

- The **-authchecking on** option enables DFS authorization checking by removing the **NoAuth** file from the machine specified with **-server**; **-authchecking off** disables authorization checking by creating the **NoAuth** file on the machine specified with **-server**.
- The **-noauth** option directs **bos** 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
```

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 *OS/390 DCE Command Reference* for more information about the **dcecp keytab** command.)

Server Accounts

DFS servers communicate across the network and run under their own network identity (**dfs-server**). To run under their own identity, servers must be programmed to perform a login and authenticate that identity. Therefore, you must create registry accounts for these servers.

Passwords for Server Accounts

During login, all principals (human, server, and machine) must enter their password in the local host. The most common method for human users is to simply type in their password. A different method must be provided for server principals. The recommended method (based on APIs supplied with DCE) is to store server passwords in a locally protected key table. The default implementation of the DCE-supplied API stores the key table in a keytab file and protects the file so that only the principal's local identity can read or write the file. Do not make this file available for access across the network.

Except for servers running as root or running under the identity of the local machine, a separate keytab file needs to be used for each server. During login, the server can access this file to obtain its password, log in, and be authenticated.

Use the Registry Editor **dcecp keytab add** subcommand to add passwords for servers to a keytab file and **dcecp keytab delete** to delete server passwords. Use the **dcecp keytab show** command to list entries in the keytab file.

If OS/390 UNIX DFS Daemons' Passwords Expire: The DFS daemons normally change their passwords automatically before the passwords expire. However, this can only happen if the daemons are running. If the password expires while the daemon is not running, that daemon cannot be restarted.

In this case, you will have to perform the following steps:

1. Use the following **dcecp** command to change the password of the IBM OS/390 DCE Base Services daemons in the Security registry.

   ```
   dcecp> account modify account_name -mypwd current_PW -password new_PW
   ```

   The principal name of the DFS servers is **hosts/***host-name***/dfs-server**.

2. Use the **dcecp keytab add** command to enter the new password in the keytab file.

For more information on the **dcecp** commands, see *OS/390 DCE Administration Guide*.

Chapter 7. Starting and Stopping the DFS Server in OS/390 DFS

This chapter briefly describes the ***DFS*** server address space, then discusses the various ways of starting and stopping DFS OS/390 daemons.

DFS Server Address Space

In DFS, the DFS server daemons run as individual processes in the DFS server address space as illustrated in Figure 8.

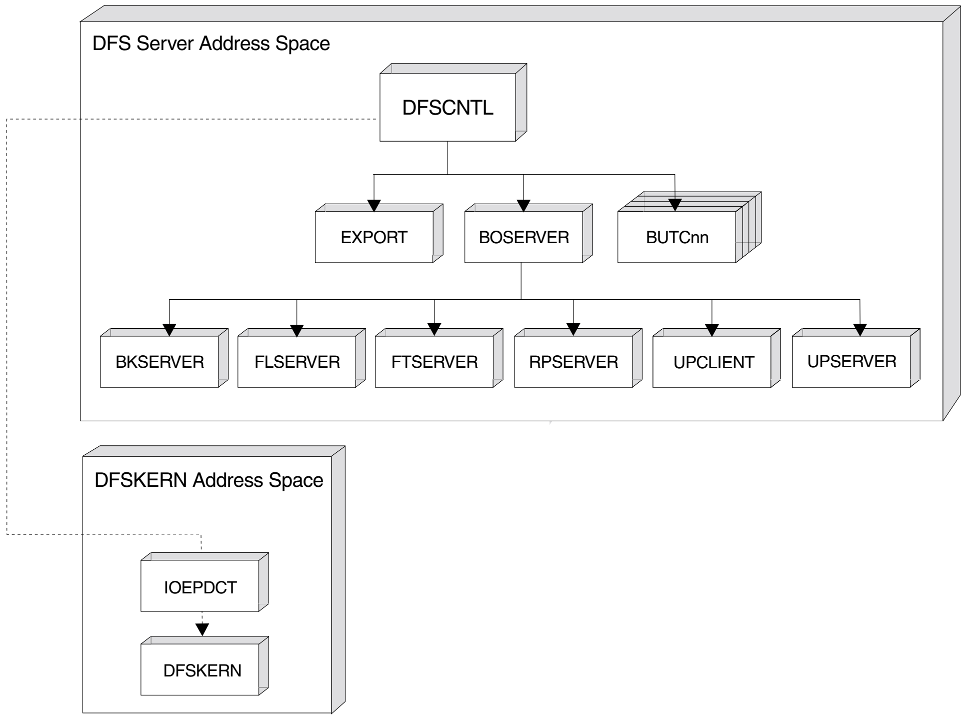


Figure 8. DFS Address Space

Shown in this figure is **dfscntl** (the DFS Control Task) that acts as the ***parent process*** to all the DFS server daemons. The DFS server daemons (**dfskern**, **export**, **boserver**, and **butc***nn*) run as child processes of **dfscntl**. DFSKERN can be run in the DFS Server Address Space or in its own address space. This is controlled by the **dfscntl** environment variable, **_IOE_DAEMONS_IN_AS**. If this environment variable is not specified, DFSKERN runs in the DFS Server Address Space (as in previous releases). If it is specified as **_IOE_DAEMONS_IN_AS=DFSKERN**, DFSKERN runs in it's own address space (called the DFSKERN Address Space). Running DFSKERN in it's own address space may reduce contention for resources and provide better failure recovery. IBM recommends that DFSKERN be run in its own address space. **MODIFY** commands are unchanged whether DFSKERN runs in its own address space or not. If you want to change where DFSKERN runs, add or remove the **dfscntl** environment variable, **_IOE_DAEMONS_IN_AS=DFSKERN**, stop DFS if it is running, and then restart DFS.

The child processes of the ***boserver*** are:

- **bkserver**
- **flserver**
- **ftserver**

- **rpserver**
- **upserver**
- **upclient**.

Important Note to Users

In DFS, there are two types of processes: **simple** and **cron**. A **simple** process is defined as a continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes. A **cron** process refers to a process that runs independently of any other processes; however, unlike a **simple** process, a **cron** process runs periodically, not continuously.

In OS/390 DFS, **simple** processes are not identified by complete pathnames to the binary files for the processes. All **simple** process commands in OS/390 DFS are members of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) created during installation (for further information, refer to the *OS/390 Program Directory*). As all OS/390 member names are limited to eight or less characters, the **repserver**, **bakserver**, and **bosserver** process member names are shortened to **rpserver** and **boserver**, respectively.

After the DFS server address space is configured, all the daemons will be started automatically when the DFS server address space is started. Boserver subprocesses are in turn started by the Boserver process.

All requests to DFS are directed to **dfscntl** that performs the requested action. The DFS server daemons can be started and stopped using an operator command. In starting and stopping the daemons, **dfscntl** uses a **Daemon Configuration File** that contains information on the daemons that were previously configured on the host. The Daemon Configuration File contains runtime options, startup parameters, and restart information for its subprocesses. For details on the Daemon Configuration File, see “Daemon Configuration File” on page 123. The Boserver process uses the BosConfig file to bring up its subprocesses. For details on the Boserver's BosConfig file, see “Process Entries in the BosConfig File” on page 132.

Besides starting and stopping the DFS server daemons, **dfscntl** can also detect daemons that have prematurely stopped and tries to restart them automatically. The algorithm used by **dfscntl** in starting and restarting the DFS server daemons is summarized in “How dfscntl Starts the DFS Server Daemons” on page 124.

The following are the PDS member names for the DFS processes started by **dfscntl**:

dfskern	The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.
export	The **export** process is started by **dfscntl** and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.
boserver	The **boserver** daemon load module name. The name, **boserver**, is an alias for the load library entry, **IOEBOSRV**.
butc*nn*	The **butc***nn* daemon load module name. There are eight valid load module names for the butc processes: **butc01**, **butc02**, **butc03**, **butc04**, **butc05**, **butc06**, **butc07**, and **butc08**. All are valid aliases for the single load module used for all **butc***nn* processes- **IOEBUTC**.

The following are the PDS member names for the DFS processes:

bkserver	Starts the Backup Server process. The name, **bkserver**, is an alias for the load library entry, **IOEBKSRV**. For further information, see “bakserver” on page 473.

flserver Starts the Fileset Location Server process. The name, **flserver**, is an alias for the load library entry, **IOEFLSRV**. For further information, see “flserver” on page 605.

ftserver Starts the Fileset Server process. The name, **ftserver**, is an alias for the load library entry, **IOEFTSRV**. For further information, see “ftserver” on page 720.

upclient Starts the Update Client process. The name, **upclient**, is an alias for the load library entry, **IOEUPCLN**. For further information, see “upclient” on page 765.

upserver Starts the Update Server process. The name, **upserver**, is an alias for the load library entry, **IOEUPSRV**. For further information, see “upserver” on page 768.

rpserver Starts the Replication Server process. The name, **rpserver**, is an alias for the load library entry, **IOERPSRV**. For further information, see “repserver” on page 745.

Who Can Start and Stop DFS Server Daemons?

There are two types of users who can start or stop the DFS server daemons in DFS:

- A user with OS/390 operator privileges.
- A user who has update privilege to the **DFSKERN.START.REQUEST** RACF* facility. This facility is created during the installation of DFS. For more information on this RACF facility, refer to the *OS/390 Program Directory*.

Starting DFS Server Daemons

DFS server daemons are started in any of the following ways:

- During the system IPL
- Using the **MODIFY DFS** operator command
- Using the **START** operator command.

Based upon the control files for **dfscntl** and the BOS Server, set up by the DFS Administrator, all of the DFS server daemons are automatically started when the DFS started task is started.

The OS/390 DFS daemons all run in one address space which is a started task (default name DFS). A user with OS/390 operator privileges can start and stop the DFS started task. The DFS server address space may be started automatically during system IPL (after DCEKERN is started). To start the DFS server address space, use the OS/390 **START** command. For example,

```
start dfs
```

After all processes are started, there is a period of time before a DFS client can access data in the DFS server due to token state recovery being processed by the File Exporter. For more information on token state recovery, see “Token State Recovery” on page 67.

Ideally, the daemons run continuously in the background and do not need to be started or stopped again. However, the DFS server daemons may have to be started manually in certain situations, for example, if a daemon ends abnormally. You can use the **MODIFY** operator command to manually start or stop the DFS server daemons.

You can also have the DFS server daemons started automatically during the system IPL.

Each of these alternatives is discussed in the following sections.

The MODIFY DFS Operator Command

The DFS server daemons (processes) can be started or stopped using the **MODIFY DFS** operator command. Using **MODIFY DFS**, you can also view the status of the DFS server daemons.

Following is the syntax of the **MODIFY DFS** command:

MODIFY DFS,*command daemon[,options]*

where

DFS	Is the name of the DFS server address space.
command	Is the action that is to be performed on the DFS sever daemon or daemons. It can have any of the following values:

start	Starts the DFS server daemon or daemons.
stop	Stops the DFS server daemon or daemons.
query	Displays the state of the DFS server daemon or daemons.
send	Sends requests to the DFS server daemon or daemons.

daemon	Is the name of the DFS server daemon for which the action is being requested. It can have any of the following values:

dfskern	DFS kernel program
export	Export program to make aggregates available for exporting
unexport	Unexport program to unexport aggregates
boserver	Basic OverSeer Server to monitor all server processes on the local machine and restart any failed processes automatically
butc*nn*	Backup Tape Coordinator that controls the behavior of the backup tape drive and accepts requests from the bak program. *nn* is a numeric value, 01 through 08.
all	All the DFS server daemons

options	Values that are passed to the daemons.

Using MODIFY DFS to Start DFS Server Daemons:

With the **MODIFY DFS** operator command, you have the option of starting an individual daemon or starting all the daemons using a single command.

For example, to start the **BUTC01** daemon, enter the following:

```
modify dfs,start butc01
```

To start all the daemons, enter the following:

```
modify dfs,start all
```

Note: Do not use the **MODIFY** command to start the DFS server daemons while the DFS server address space is still initializing. During initialization, DFS will attempt to start all the DFS server daemons that have been configured on the OS/390 host. If you issue the **MODIFY** command while DFS is initializing, the DFS server daemons may be started out of order or stopped erroneously. This may lead to unexpected errors during initialization and cause DFS to end abnormally.

It is recommended that you wait until DFS has issued a log message indicating that DFS server initialization has completed before using the **MODIFY** commands.

Order of Starting DFS Server Daemons

When DFS server daemons are started manually, the successful startup of some daemons depend on the availability of the services provided by other daemons. This implies that the DFS server daemons must be started in a particular order.

Before you can start up any of the DFS server daemons, the DCE Security daemon and the DCE CDS daemon must already be running in the cell.

Following is the sequence by which DFS server daemons should be started.

Note: This is only applicable if you need to start any of the DFS server daemons individually. If the DFS server daemons are started collectively, (for example, using the **start all** option of the **MODIFY DFS** command) DFS ensures that the correct starting sequence is followed.

1. **dfskern**
2. **export**
3. **boserver**

For example, to successfully start the **boserver** daemon, the **dfskern** daemon must already be up and running.

Note: Make sure that the passwords of the DFS server daemons are valid before starting them up. If the passwords have expired, the daemons cannot be started. For more information on daemon passwords, see "Passwords for Server Accounts" on page 116.

Stopping DFS

To stop the DFS server address space, use the **STOP** operator command to ensure the normal shutdown of the address space.

To stop the DFS server address space and all DFS server daemons, enter the following OS/390 operator command:

```
stop dfs
```

To stop the DFS server daemons, but not the DFS server address space, use the **STOP ALL** command. For example:

```
modify dfs,stop all
```

The **STOP ALL** command causes **dfscntl** to stop all daemons that it controls. The BOS Server is stopped, causing all daemons controlled by it to be stopped.

Using MODIFY DFS to Stop DFS Server Daemons

You can use the **MODIFY DFS** system command to stop a DFS server daemon or all daemons that are configured on the host.

For example, to stop the boserver daemon, enter:

```
modify dfs,stop boserver
```

To stop all DFS server daemons on the host, enter:

```
modify dfs,stop all
```

Viewing the Status of DFS Server Daemons

You can query the status of the DFS server daemons using the **query** option of the **MODIFY** system command. You do not need the special privileges of a DCE or DFS administrator or an operator to use the **QUERY** option.

For example, to query the status of the **dfskern** daemon, enter the following:

```
modify dfs,query dfskern
```

A message about the status of the daemon will be written on the system log. This message will also contain the **process ID** of the daemon.

The status of the daemon can be any of the following:

READY	Indicates that the daemon is running, has been initialized, and is ready to receive and process incoming requests.
ACTIVE	Indicates that a manual process is running. When an active process stops, it is never considered an error and it is never restarted automatically.
INITIALIZING	Indicates that the daemon has been started, but is not yet ready to receive and process incoming requests.
STOPPING	Indicates that a request to stop the daemon has been received and that the daemon is in the process of stopping.
DOWN	Indicates that the daemon is not active.
REGISTERED	Indicates that the daemon has registered with the DFSCNTL task and has started. This means that the process is either initializing or is ready. This state only applies to sub-processes of the BOSERVER.
UNKNOWN	Indicates that the status of the daemon cannot be determined. This can occur if the daemon was started, but no response was received by the system indicating a change in its status. **Note:** You can issue a command to stop a daemon if it is in the UNKNOWN state.

The status of OS/390 DFS daemons controlled by **dfscntl** can be queried by OS/390 operator commands. For example:

```
modify dfs,query all
modify dfs,query dfskern
modify dfs,query boserver
modify dfs,query butc01
```

Following is an example of a query command to **dfscntl** The output is sent to the OS/390 Operator's console:

```
modify dfs,query dfskern
```

```
IOEP00022I DFS daemon DFSKERN status is READY and process id is 781.
```

Starting DFS Server Daemons During System IPL

Because the DFS server daemons are contained in the DFS server address space, these daemons are started during the initialization of DFS. This allows you to configure the host to automatically start the DFS server address space during the system IPL.

dfscntl uses the Daemon Configuration File to determine which daemons can be started, and the parameters to pass to the daemon load module when starting the daemon. The DFS server address space may be started automatically during system IPL (after DCEKERN is started). To start the DFS server address space, use the OS/390 **START** command. For example,

```
start dfs
```

Daemon Configuration File

The Daemon Configuration File is used by **dfscntl** to obtain necessary information when starting the DFS server daemons. The Daemon Configuration File contains the following information:

- Parameters that are passed to the load module when a daemon is started, called the argument list (including LE/370 runtime options).
- The **Minimum Restart Interval**. **dfscntl** attempts to restart a daemon that ends abnormally only if the daemon was running for at least this time interval. If a daemon ends during this time interval, it will not be restarted.
- The **Time-out Period**, which is the maximum time interval that **dfscntl** waits for the daemon to complete its initialization after it has been started. When this time interval elapses, and **dfscntl** has not received confirmation from the daemon that initialization has completed, the status of the daemon is set to **UNKNOWN**.

Note: Restart and Timeout do not apply to the sub-processes of the BOSERVER.

The pathname to the Daemon Configuration file is **/opt/dfslocal/etc/ioepdcf**. Figure 9 shows the typical contents of the Daemon Configuration file.

```
DFSKERN   CONFIGURED=Y LMD=DFSKERN  ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern')/-admingroup subsys/dce/dfs-admin>DD:DFSKERN2>&1"
    RESTART=300 TIMEOUT=300
EXPORT    CONFIGURED=I LMD=IOEDFSXP ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfsexport')/-all -verbose     >DD:EXPORT   2>&1"
    RESTART=300 TIMEOUT=300
UNEXPORT CONFIGURED=M LMD=IOEDFSXP ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfsexport')/-detach -all -ver >DD:UNEXPORT2>&1"
    RESTART=300 TIMEOUT=300
| BOSSERVER CONFIGURED=N LMD=BOSERVER ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/boserver')/     >DD:BOSERVER 2>&1"
    RESTART=300 TIMEOUT=300
BUTC01    CONFIGURED=M LMD=BUTC01   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc01')/ -tcid 0 >DD:BUTC01   2>&1" RESTART=300 TIMEOUT=300
BUTC02    CONFIGURED=M LMD=BUTC02   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc02')/ -tcid 1 >DD:BUTC02   2>&1" RESTART=300 TIMEOUT=300
BUTC03    CONFIGURED=M LMD=BUTC03   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc03')/ -tcid 2 >DD:BUTC03   2>&1" RESTART=300 TIMEOUT=300
BUTC04    CONFIGURED=M LMD=BUTC04   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc04')/ -tcid 3 >DD:BUTC04   2>&1" RESTART=300 TIMEOUT=300
BUTC05    CONFIGURED=M LMD=BUTC05   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc05')/ -tcid 4 >DD:BUTC05   2>&1" RESTART=300 TIMEOUT=300
BUTC06    CONFIGURED=M LMD=BUTC06   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc06')/ -tcid 5 >DD:BUTC06   2>&1" RESTART=300 TIMEOUT=300
BUTC07    CONFIGURED=M LMD=BUTC07   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc07')/ -tcid 6 >DD:BUTC07   2>&1" RESTART=300 TIMEOUT=300
BUTC08    CONFIGURED=M LMD=BUTC08   ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc08')/ -tcid 7 >DD:BUTC08   2>&1" RESTART=300 TIMEOUT=300
```

Figure 9. Daemon Configuration File

In the **ARG** (argument list) field of this example, **ENVAR** indicates the environment variables to be used, in this case, the **_EUV_HOME** environment variable which points to the daemon's home directory. Anything after the “/” character are program parameters. The “>” character is the redirection character which indicates that the output will be redirected to the DD name that follows.

Important Note to Users

The default **/opt/dfslocal/etc/ioepdcf** file created by DFS post initialization processing by **dfs_cpfiles** is set up to only start **dfskern** when the DFS server address space is next initialized. It is recommended that you update the **ioepdcf** file to automatically start the **boserver** by setting the BOSERVER entry to CONFIGURED=Y before starting DFS.

Under normal circumstances, no further modifications of the Daemon Configuration File are necessary. Although there may be other situations when the Daemon Configuration File has to be modified, it is recommended that you do so under the supervision of an IBM service representative.

The Daemon Configuration File can only be modified when the DFS server address space is not running.

How dfscntl Starts the DFS Server Daemons

When a request arrives to start DFS server daemons, **dfscntl** looks at the Daemon Configuration File to see if the particular daemon or daemons are configured on the host. If the daemon is configured and is not running, **dfscntl** starts it and waits for the daemon to initialize successfully.

In case of the abnormal ending of any DFS server daemon, **dfscntl** tries to restart the daemon. **dfscntl** attempts to restart the daemon only if the daemon was running for at least the duration of the Minimum Restart Interval, as specified in the Daemon Configuration File. If the daemon ended within this time interval, it will not be restarted.

Note: When **dfscntl** restarts an abnormally terminated daemon, it does not correct the problem that caused the daemon to end unexpectedly. Thus, depending on the cause of the abnormal ending, the daemon may fail again after restarting because of the same error condition.

Using the -nodfs Option to Start dfscntl

You can start the DFS server address space without starting the configured DFS server daemons by using the **-nodfs** option of the **START** operator command, for example:

```
start dfs,parm='-nodfs'
```

Chapter 8. Starting and Stopping the DFS Cache Manager in OS/390 DFS

This chapter discusses the DFS Cache Manager (**DFSCM**) address space and various ways of starting and stopping the DFS client.

DFSCM Address Space

The DFS client runs are implemented as a physical file system that runs in the colony address space named **DFSCM**. The **DFSCM** address space is started when OMVS is initialized. Refer to the *OS/390 UNIX System Services File System Interface Reference*, SC28-1909, for more information on physical file systems and colony address spaces.

The DFS client daemons **ioecmini**, **ioedfsd**, and **ioelogin** run as individual processes in the **DFSCM** address space as shown in Figure 10.

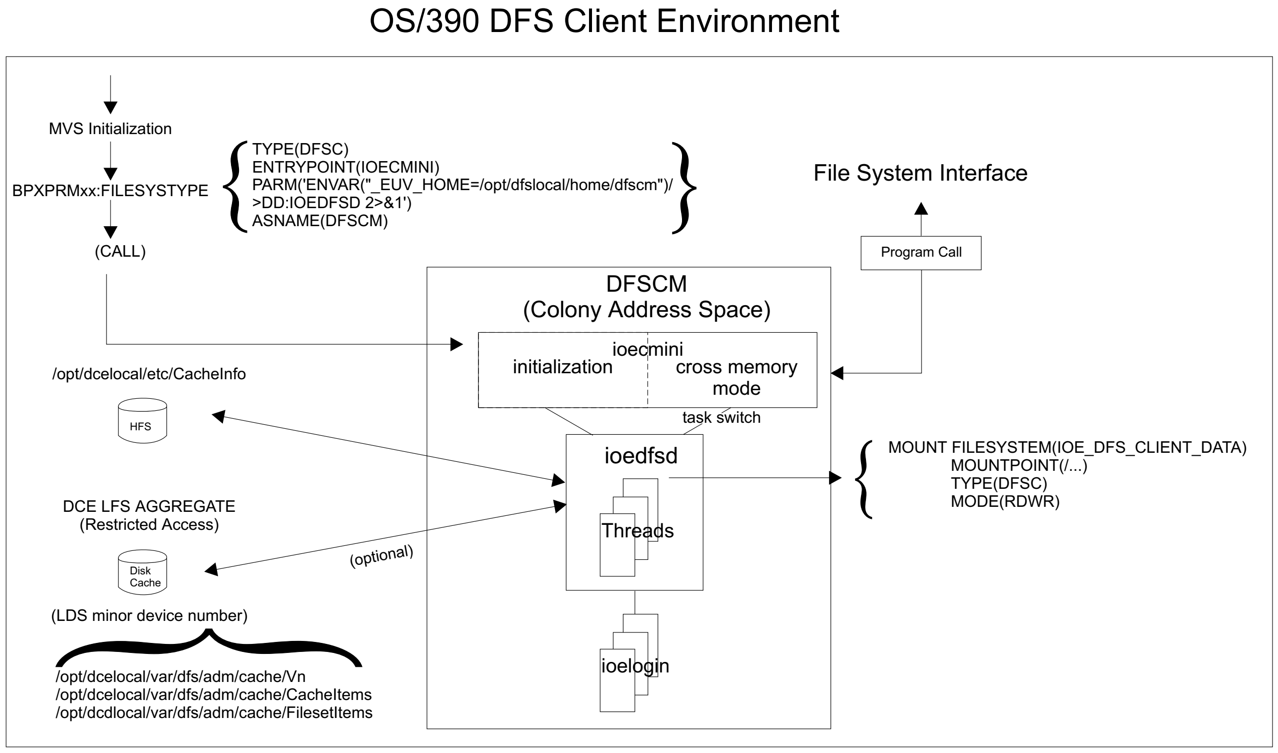


Figure 10. DFSCM Address Space

Refer to “How OS/390 UNIX Starts the DFS Cache Manager” on page 127 for information on **DFSCM** initialization processing.

Important Note to Users

The **ioedfsd** process is also unique in OS/390 in that it combines the functions of the traditional Cache Manager **dfsd** and **dfsbind** processes. In OS/390, references to the **dfsd** or **dfsbind** process should be taken to mean the **ioedfsd** process.

Who Can Start and Stop DFSCM?

A user with OS/390 operator privileges is the only one who can start or stop the DFS Cache Manager (**DFSCM**).

Starting DFS Cache Manager (DFSCM)

DFSCM is started automatically.

- During the system IPL as part of OMVS initialization.

  Note: DFSCM waits for DCE to be installed.

- After the initialization has started successfully you receive the following message:

  ```
  IOEC041831 dfsd: initialization complete
  ```

In OS/390, you can customize and control several DFS client features by specifying **DFSCM** initialization parameters in the CacheInfo file and the **_IOE_CM_PARMS** environment variable.

Stopping and Restarting DFSCM

To **stop** the **DFSCM** address space, use the **stop** operator command. (Refer to the *OS/390 UNIX System Services Command Reference*, SC28-1892, for more information on the **stop** command.)

```
stop dfscm
```

To **restart** the **DFSCM** address space, reply to the operator console message:

```
*nn BPXF014D FILESYSTYPE DFSC TERMINATED.  REPLY 'R' WHEN READY TO RESTART.
```

Note: DCE must be running to restart DFSCM.

Starting DFSCM During System IPL or OMVS Restart

DFSCM is a colony address space that is automatically started by OMVS as a physical file system during system IPL when OMVS is started or when OMVS is restarted. The DFS client is identified as a physical file system in the **BPXPRM***xx* **FILESYSTYPE TYPE(DFSC)** parmlib entry.

DFSCM daemons **ioecmini**, **ioedfsd**, and **ioelogin** run as processes within the **DFSCM** address space and cannot be controlled separately.

DFSCM Configuration Files

In OS/390, the following list describes how you can customize and control several DFS client features by specifying **DFSCM** initialization parameters:

- Local HFS **/opt/dcelocal/etc/CacheInfo** file

  A parameter specified in the **CacheInfo** file is overridden by the same parameter specified in the **envar** file.

- **_IOE_CM_PARMS** environment variable

  Parameters specified in the **_IOE_CM_PARMS** environment variable file override equivalent parameters specified in the **CacheInfo** file.

The DFS client initialization parameters are described as **dfsbind** command options in “dfsbind” on page 586 and as the **dfsd** command options in “dfsd” on page 589. The **CacheInfo** file is described in “CacheInfo” on page 363 and the **envar** file is described in “DFS Client ENVAR File” on page 274. Also, for more information on the **BPXPRM***xx* entry for the DFS client refer to “How OS/390 UNIX Starts the DFS Cache Manager” on page 127.

Important Note to Users

DFSCM information specified in the **BPXPRM***xx* parmlib only takes effect for **DFSCM** when OMVS is started. The **DFSCM _EUV_HOME** environment variable must be specified in the **BPXPRM***xx* **FILESYSTYPE TYPE(DFSC)** parmlib entry.

How OS/390 UNIX Starts the DFS Cache Manager

In OS/390, the DFS client runs as the colony address space DFSCM. When OMVS finds a **FILESYSTYPE TYPE(DFSC)** entry in the **BPXPRM***xx* parmlib entry, it invokes the **ioecmini** process. The **ioecmini** process performs several functions including:

- Starting the **ioedfsd** process which turn, creates various worker threads
- Mounting the DFS file system name specified in the **BPXPRM***xx* **FILESYSTYPE TYPE(DFSC)** parmlib entry for the DFS client.
- Returning to OMVS before it has completed initialization
- Waiting for DCEKERN to initialize before completing initialization.

After **DFSCM** notifies OS/390 UNIX that the DFS client is available as a physical file system, **ioecmini** is subsequently invoked by OS/390 UNIX in cross memory mode by a program call as a result of a file system interface request when an OMVS user or application references a directory or file in the DFS (**/...**) namespace. The **ioecmini** process handles the request or queues work for the **ioedfsd** process which does not run in cross memory mode. The **ioedfsd** process and its threads perform the necessary work to service the request and returns information to the **ioecmini** process which in turn, returns the information to OS/390 UNIX and the OMVS user or application. Additionally, requests are queued to the **ioelogin** process by the **ioedfsd** process to perform DCE login services.

Note: An abnormal termination in the **DFSCM** address space occurring after initialization results in an automatic restart of the **DFSCM** address space. Messages written to the system log identify the error, that an SVC dump was taken, that the **DFSCM** address space is terminating, and that an automatic restart of the **DFSCM** address space occurs.

Important Note to Users

The **ioedfsd** process is also unique in OS/390 in that it combines the functions of the traditional Cache Manager **dfsd** and **dfsbind** processes. The **ioecmini**, **ioedfsd**, and **ioelogin** processes are unique because they combine the functions of the traditional **dfsd** and **dfsbind** processes.

Chapter 9. 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 the processes it controls, 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: **dfskern, dfsd, dfsxport, dfscntl, butc***nn*. They are not intended to be controlled by the bosserver. For more information see Chapter 7, “Starting and Stopping the DFS Server in OS/390 DFS” on page 117 and Chapter 8, “Starting and Stopping the DFS Cache Manager in OS/390 DFS” on page 125 for **dfsd** process.

Important Note to Users

OS/390 DFS does not support installing process binary files onto a DFS server machine. Because installing process binary files is not supported on the OS/390 operating system, the following **bos** commands and option are not supported on the OS/390 operating system:

- **bos getdates** command
- **bos install** command
- **bos uninstall** command
- **bos prune** command
- **-newbinary** option.

Important Note to Users

In DFS, there are two types of processes: **simple** and **cron**. A **simple** process is defined as a continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes. A **cron** process refers to a process that runs independently of any other processes; however, unlike a **simple** process, a **cron** process runs periodically, not continuously.

In OS/390 DFS, **simple** processes are not identified by complete pathnames to the binary files for the processes. All **simple** process commands in OS/390 DFS are members of the *xxx*.SIOELMOD (where *xxx* is installation dependent) data set created during installation (for further information, refer to the *OS/390 Program Directory*). As all OS/390 member names are limited to eight or less characters, the **repserver**, bakserver, and **bosserver** process member names are shortened to **rpserver** and **boserver**, respectively.

The following are the PDS member names for the DFS processes started by the DFS Control Task:

boserver	The **boserver** daemon load module name. The name, **boserver**, is an alias for the load library entry, **IOEBOSRV**.
butc*nn*	The **butc***nn* daemon load module name. There are eight valid load module names for the butc processes: **butc01**, **butc02**, **butc03**, **butc04**, **butc05**, **butc06**, **butc07**, and **butc08**. All are valid aliases for the single load module used for all **butc***nn* processes- **IOEBUTC**
dfskern	The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.

In addition, the **export** process is started by the DFS Control Task and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.

The following are the PDS member names for the DFS processes:

bkserver	Starts the Backup Server process. The name, **bkserver**, is an alias for the load library entry, **IOEBKSRV**. For further information, see “bakserver” on page 473.
flserver	Starts the Fileset Location Server process. The name, **flserver**, is an alias for the load library entry, **IOEFLSRV**. For further information, see “flserver” on page 605.
ftserver	Starts the Fileset Server process. The name, **ftserver**, is an alias for the load library entry, **IOEFTSRV**. For further information, see “ftserver” on page 720.
upclient	Starts the Update Client process. The name, **upclient**, is an alias for the load library entry, **IOEUPCLN**. For further information, see “upclient” on page 765.
upserver	Starts the Update Server process. The name, **upserver**, is an alias for the load library entry, **IOEUPSRV**. For further information, see “upserver” on page 768.
rpserver	Starts the Replication Server process. The name, **rpserver**, is an alias for the load library entry, **IOERPSRV**. For further information, see “repserver” on page 745.

About the DFS Server Processes

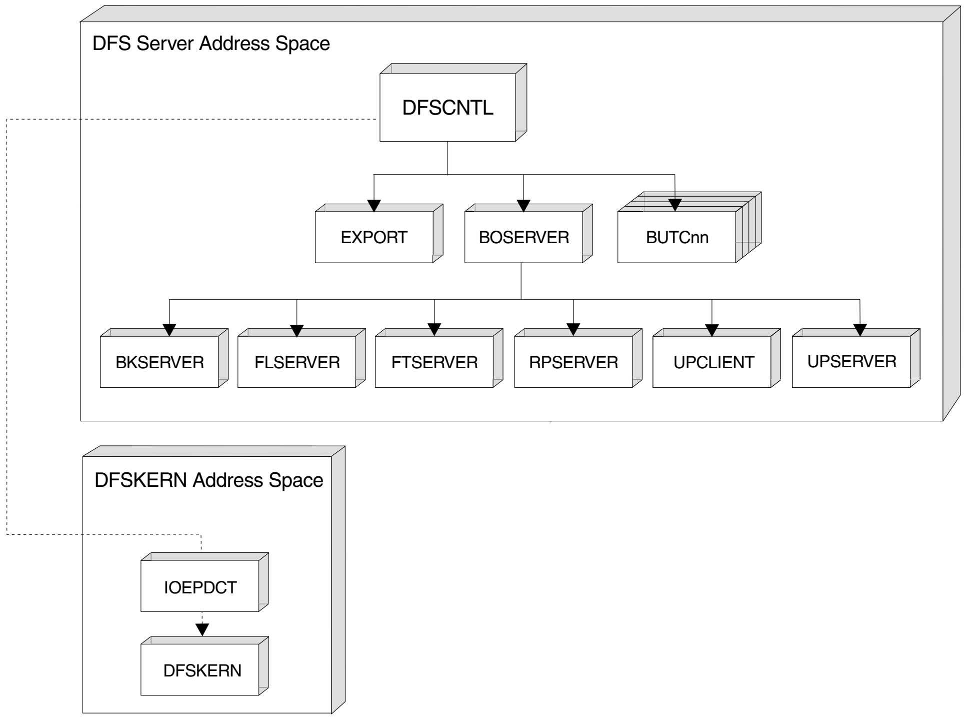


Figure 11. DFS Address Space

The DFS Control Task is the parent process to all other DFS server processes. Figure 11 shows the DFS Control Task and its subprocesses. When a user submits a request to start DFS and its processes, the following happens:

- The DFS Control Task looks at the **/opt/dfslocal/etc/ioepdcf** file to determine which DFS server processes to start up.

  /opt/dfslocal/etc/ioepdcf contains the names of each process to be started. Along with the name of each process is a configured parameter that specifies actions to be taken when the process is started. The configured parameter can have the following values:

Y	Start the process during initialization. If the process ends abnormally, then restart it automatically.
N	Do not start the process automatically or manually.
I	Start the process during initialization. The process can be started manually. If the process ends, it will not restart.
M	Can be started manually.

  The following processes may be specified in the **ioepdcf** configuration file:

  – **dfskern**
  – **export**
  – **unexport**
  – **boserver**
  – **butc***nn* where *nn* can be a number from 01 to 08.

- There are six subprocesses under the Boserver. The Boserver can start and stop these subprocesses. When the Boserver starts, it will start up any of these subprocess specified by the **bos create** command. The processes that the BOSERVER can start and stop:

  bkserver Backup server to perform dump and restore operations.

flserver	Fileset Location Server to track the location of the filesets.
ftserver	Fileset server to export and store DCE Local File System and non-Local File System data.
rpserver	Replication server to replicate filesets.
upserver	Update server to store and distribute configuration files (administration lists).
upclient	Update client to retrieve configuration files (administration lists) from the Upserver.

Process Entries in the BosConfig File

You define which processes the BOS Server monitors by creating **process entries** in the **/opt/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:

- **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.
- **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 server 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:

  1. The command that is to be executed;
  2. The time that the command is to be executed.

- **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.

  Note: 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.

Note: The **NotRun** status flag appears in the **BosConfig** file as a 0.

- **command parameters** — These parameters are used by the BOS Server to run the process.

The following output from the **bos status** command displays an entry from the **BosConfig** file. (See "Listing Status and Machine Information" on page 137 for more information about the **bos status** command, which is used to list entries from the **BosConfig** file.)

```
$ bos status /.:/hosts/RANDOM ftserver -long
Instance ftserver, (type is simple) currently running normally.
    Process last started at Fri Apr 26 12:01:30 1996 (2 proc starts)
    Last exit at Fri Apr 26 11:25:11 1996
Parameter 1 is 'ftserver envar('_EUV_HOME=/opt/dfslocal/home/ftserver'),/>
dd:ftserver 2>&1 -verbose'
```

It is possible for the BOS Server's memory state to change independently of the **BosConfig** file. The BOS Server checks the **BosConfig** file whenever it starts or restarts, (in response to the **bos restart** command, at the general restart time, or at system reboot). At that time, the BOS Server transfers information from the file into memory and does not read the file again until it restarts.

Therefore, it is possible to use the **bos shutdown** command to stop a running process, even though its status flag in the **BosConfig** file is **Run**. Similarly, you can use the **bos startup** command to start a process running by setting its memory state status flag to **Run** without setting its status flag in the file to **Run**. The commands discussed in this chapter can affect the BOS Server's memory state, the information in the **BosConfig** file, or both.

Starting or stopping certain processes, either temporarily or permanently, has an effect on the other processes that run on the other server machines in your cell. For example, an **upserver** process must run on each System Control machine and Binary Distribution machine. If you start or stop the process on one machine, you must start it on a replacement System Control or Binary Distribution machine. You must also modify the **upclient** processes on the appropriate server machines so that they reference the new System Control or Binary Distribution machine.

Standard Information in this Chapter

The following sections present options and arguments common to many of the commands described in this chapter. It also presents some common operations that are explained in the chapter or that can be useful when performing other operations.

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See Chapter 19, "Distributed File Service Commands" on page 401 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.

  If you want 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 File Server machine using the full DCE pathname, for example, **/.../abc.com/hosts/fs1**.

If you want to run an unprivileged **bos** command, you can use any of the following to specify the File Server machine:

- The machine's DCE pathname (for example, **/.../abc.com/hosts/fs1**) and the **-noauth** option
- The machine's hostname (for example, **fs1.abc.com** or **fs1**)
- The machine's IP address (for example, **11.22.33.44**).

Note: If you run a **bos** command and specify the File Server machine with either the machine's hostname or IP address, the command executes using the unprivileged identity **nobody** (the equivalent of running the command with the **-noauth** option). To inform you of this situation, the command displays the following warning message (unless the **-noauth** flag is used to suppress it):

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

Unless the corresponding File Server machine is in noauth mode, running a privileged **bos** command in this manner causes it to fail.

- 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:

bakserver	The Backup Server process
flserver	The Fileset Location Server process
ftserver	The Fileset Server process
repserver	The Replication Server process
upserver	The server portion of the Update Server
upclient	The client portion of the Update Server that transfers configuration files (such as administrative lists) from **/opt/dcelocal/var/dfs** on the System Control machine.

- The **-cmd** *cmd_line* option specifies the commands and parameters that the BOS Server uses. 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 **"/opt/dfsglobal/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. Standard OS/390 program searching is used by the OS/390 UNIX **attach_execmvs** callable service to find the program.

  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 6, “Using Administrative Lists and Keytab Files” on page 103 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 10, “Making Filesets and Aggregates Available” on page 147 for information about DFS server principals.)

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

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

Standard Commands and Operations

Some of the following commands and operations are described in many places in this chapter; others can prove useful when the operations in this chapter are performed. If a command or operation is described in detail here, it is not described in depth in later sections of this chapter where it is used.

Determining Administrative Privilege

To perform the majority of **bos commands**, the issuer must be listed in the **admin.bos** file on the machine used in the command. To determine the members of a list, issue the **bos lsadmin** command:

```
$ bos lsadmin -server machine -adminlist admin.bos
```

The **-adminlist admin.bos** option specifies that members of the **admin.bos** file are to be listed.

Examining Log Files: The **bosserver** process and most of the server processes it monitors generate log files. The log files record execution messages and error messages generated by the server processes as they execute. By default, the processes write the files to the **/opt/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 **CMLog** file is generated by the Cache Manager on each client machine.
- The **DfsgwLog** file is generated by the Gateway Server process on each Gateway Server machine.

Important Note to Users

In OS/390 DFS, the **DfsgwLog** file is not available. See Appendix D, “DFS/NFS Secure Gateway” on page 805 for more information on the DFS/NFS Secure Gateway.

- 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 **/opt/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 file name is sufficient for a log file that resides in the **/opt/dcelocal/var/dfs/adm** directory. A full pathname is required for a log file that resides in a different directory.

Creating and Starting Processes

To start a new process on a server machine, use the **bos create** command to alter the **BosConfig** file. This adds a process entry for the new process to the **BosConfig** file and sets the status flag for the process to **Run** in both the file and the BOS Server's memory, making the effect immediate. You can use the command to create both **simple** and **cron** processes.

The server process name included in this command is used by the BOS Server to reference the process. It is also used in any subsequent **bos** commands that require a process name. The BOS Server adds it to the **BosConfig** file when it creates the process's entry. The name does not appear in process listings generated with the **ps** command or its equivalent.

Creating and Starting a simple Process

To create and start a **simple** process, perform the following steps:

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 **ftserver** on the machine named **fs1**:

```
$ bos create /.:/hosts/RANDOM ftserver simple \
"ftserver envar('_euv_home=/opt/dfslocal/home/ftserver')/ >dd:ftserver"
```

In this example, **/.:/hosts/RANDOM** identifies the OS/390 DFS host running the DFS server started task, while **ftserver simple** specifies the server process name and process type. The portion of the command in double quotes specifies the program **ftserver** to be executed, a C runtime parameter setting an environment variable, and redirects STDOUT for this process to DDNAME FTSERVER. The slash (/), just before >dd:ftserver, separates the C runtime parameters from the parameters passed to the program.

Creating and Starting a cron Process

To create and start a **cron** process, perform the following steps:

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 **aixfs1**. The **-localauth** option allows the unauthenticated process to use the DFS server principal of **aixfs1** to execute the privileged **fts clonesys** command.

```
$ bos create /.../abc.com/hosts/aixfs1 backup cron \
"IOEFTS clonesys -s /.../abc.com/hosts/aixfs1 -localauth" 5:30
```

For OS/390 DFS, the pathname for the program to be executed is not needed, instead the PDS member name is used. In the following example, the **fts clonesys** command is run at 5:30 am.

```
$ bos create /.:/hosts/mvsfs1 backup cron \
"IOEFTS clonesys -s /.:/hosts/mvsfs1 -localauth" 5:30
```

Listing Status and Machine Information

Use the **bos status** command to check the processes that are running on a server machine. The command causes the BOS Server to probe and determine the status of each process on the machine. It then displays output about the status of each process. It also displays appropriate messages if DFS authorization checking is disabled on the machine or if the machine's **/opt/dfslocal** directory or its contents are not protected appropriately.

The process information provided by the **bos status** command enables you to determine the role of the server machine (File Server machine, System Control machine, Binary Distribution machine, Fileset Database machine, Backup Database machine, or multiple machine roles). When you are using the **bos status** command to determine server machine roles, include the **-long** option to provide more detailed output.

Checking the Status of Processes on a Server Machine

Enter the **bos status** command to check the status of the processes on a server machine. Use the **-process** option to display the status of specific processes on the specified server machine, or omit the option to display the status 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.

Interpreting the Output

The command first displays the following line if DFS authorization checking is disabled on the machine (it does not display the line if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

It then displays the following line if the BOS Server finds that the **/opt/dfslocal** directory or a directory or file beneath it on the machine has protections that it believes are inappropriate:

```
Bosserver reports incorrect access on server directories.
```

The BOS Server displays the message if the UNIX mode bits on the **/opt/dfslocal** directory and its contents do not enforce certain protections. The message usually indicates that users who should not be able to write to the **/opt/dfslocal** 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 "bos status" on page 531 for the description of the **bos status** command and the protections the BOS Server wants to see enforced.)

The command then displays status information about the processes on the machine. The possible status 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 **/opt/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 starts** occurred (proc starts occur when the process is started or restarted by the current BOS Server).
- The time of the last proc start.
- The exit time and error exit time when the process last failed. This appears only if the process failed while the BOS Server was running. (Provided the BOS Server was running both when the process was started and when it failed, the BOS Server can provide this information for any process that has an entry in the **BosConfig** file.)
- The command and its options 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 'opt/dfsglobal/bin/ftserver'
```

Determining Server Machine Roles

The following instructions can help you use the **bos status** command to determine which server machines are filling the various machine roles in your cell or domain. The instructions assume that your cell is configured according to the installation and configuration instructions for your system; for example, they assume that all machines except the System Control machine are running a client portion of the Update Server that references the **/opt/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, **/opt/dcelocal/var/dfs**.

Important Note to Users

OS/390 DFS cannot act in a binary distribution role on either client or server.

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/hosts/fs1 upserver -long
```

```
bos: unable to get instance info for 'upserver' (no such entity)
```

The output for a System Control machine includes references to the **upserver** process and the **/opt/dcelocal/var/dfs** directory, where administrative lists are stored.

```
$ bos status /.../abc.com/hosts/aixfs2 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 '/opt/dceglobal/bin/upserver dcelocal/var/dfs'
```

Stopping and Removing Processes

You can stop a process by using the **bos stop** command to set its status flag to **NotRun** in both the BOS Server's memory and the **BosConfig** file. The process then appears as `disabled` in the output from the **bos status** command. The entry remains in the file, but it does not run again until you issue the **bos start** command, which changes its status flag to **Run** in *both* the memory and the **BosConfig** file. You can also issue the **bos startup** command to run the process by changing its status flag *only* in memory.

To halt a process temporarily (for example, to perform maintenance or make alterations to a configuration), use the **bos shutdown** command to change the process's status in the BOS Server's memory to **NotRun**. The effect is immediate and remains until you again change the memory state or until the BOS Server restarts, at which time it consults the **BosConfig** file and sets the memory state to match the information in the file.

After you stop a process with the **bos stop** command, you can remove it from the **BosConfig** file with the **bos delete** command. It then no longer appears in the output from the **bos status** command. You must use the **bos stop** command to stop a process (**simple** or **cron**) whose status is **Run** before you use the **bos delete** command to remove it from the **BosConfig** file. An error occurs if the status of a process being deleted is **Run** when the **bos delete** command is issued.

CAUTION:
Do not temporarily stop a database server process on all machines simultaneously. This would make the database totally unavailable.

Stopping Processes by Changing Their Status Flags to NotRun

To stop processes by changing their status flags to **NotRun**, perform the following steps:

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]
   ```

   If you specify the **-wait** option, the command shell prompt will not reappear until the processes have stopped. If you do not specify the **-wait** option, the prompt will reappear immediately, even if the processes have not yet stopped.

Stopping Processes Temporarily

To stop processes temporarily, perform the following steps:

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.

If you specify the **-wait** option, the command shell prompt will not reappear until the processes have stopped. If you do not specify the **-wait** option, the prompt will reappear immediately, even if the processes have not yet stopped.

Removing Processes from the BosConfig File

To remove processes from the **BosConfig** file, perform the following steps:

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**. This must also be done for the **cron** processes, even though they do not run continuously.

   ```
   $ bos stop -server machine -process server_process...[-wait]
   ```

   If you specify the **-wait** option, the command shell prompt will not reappear until the processes have stopped. If you do not specify the **-wait** option, the prompt will reappear immediately, even if the processes have not yet stopped.
3. Remove the processes from the **BosConfig** file with the **bos delete** command:

   ```
   $ bos delete -server machine -process server_process
   ```

Starting Processes

When starting processes, you can use the **bos start** command to change their status flags to **Run** in both the **BosConfig** file and in the BOS Server's memory. You can also start processes that are temporarily disabled (processes that have a status of **Run** in the **BosConfig** file but a status of **NotRun** in memory) by using the **bos startup** command and changing only the memory state to **Run**. You can use the **bos startup** command to change a process's status in memory to **Run** even if its status in the **BosConfig** file is **NotRun**; thus, you can use the **bos startup** command to run tests on a server process without enabling it permanently.

A newly started process is a completely new instance; if you install new binaries during the time a process is shut down, they are used when you issue **bos start** or **bos startup**. If an instance of a process is already running, the only effect of these commands is to ensure that the process's status flag is set to **Run** in memory and, if **bos start** is used, in the **BosConfig** file; you must issue the **bos restart** command to start a new instance of the process.

Starting Processes by Changing Their Status Flags to Run

To start processes by changing their status flags to **Run**, perform the following steps:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos start** command to start the processes by changing their status flags in the **BosConfig** file and in memory to **Run**:

   ```
   $ bos start -server machine -process server_process...
   ```

Starting All Stopped Processes That Have BosConfig Flags of Run

To start all stopped processes that have status flags of **Run** in the **BosConfig** file, perform the following steps:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Use the **bos startup** command to start each process that has a status flag of **Run** in the **BosConfig** file; this changes each process's status flag in the BOS Server's memory from **NotRun** to **Run**. Each process's status flag in the **BosConfig** file remains the same.

   ```
   $ bos startup -server machine
   ```

Starting Specific Temporarily Stopped Processes

To start processes that were temporarily stopped, perform the following steps:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Use the **bos startup** command to start each specified process by changing its status flag in the BOS Server's memory to **Run**. Each process's status flag in the **BosConfig** file remains unchanged.

   ```
   $ bos startup -server machine -process server_process...
   ```

Restarting Processes

You may sometimes need to stop and then restart a process (for example, to load a new binary file immediately rather than wait for the BOS Server to perform its daily check for new files, which is described in "Setting Scheduled Restart Times for OS/390" on page 143. 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 restarts 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:

   a. **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...
      ```

   b. **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 the 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**.

c. **Stop and restart all processes except the BOS Server**

Omit both the **-process** and **-bosserver** options from the **bos restart** command. The BOS Server stops all processes except itself. It then immediately restarts all processes with the status flag **Run** in the **BosConfig** file.

```
$ bos restart -server machine
```

Installing Process Binary Files

Important Note to Users

OS/390 DFS does not support installing process binary files onto a DFS server machine. Because installing process binary files is not supported on the OS/390 operating system, the following **bos** commands and option are not supported on the OS/390 operating system:

- **bos getdates** command
- **bos install** command
- **bos uninstall** command
- **bos prune** command
- **-newbinary** option.

For additional **Important Note to Users** that you should be aware of, see page 130.

Setting Scheduled Restart Times for OS/390

The BOS Server performs two types of scheduled restarts: a general restart and a new binary restart. During a general restart, the BOS Server restarts itself (using a new load module, if one exists) and then it 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 **never**; by default, the BOS Server restart time is **never**.

Important Note to Users

In OS/390 DFS the binary restart time is not available.

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 or reset both time settings with the **bos getrestart** and **bos setrestart** commands.

Although the default is **never**, a general restart time can be set as a day and time or as just a time. When including a day, specify the day first, in lowercase letters; you can enter the entire name or just the first three letters; for example, **sunday** or **sun**. When indicating a time, separate hours from minutes with a colon; you can use 24-hour time or 1:00 through 12:00 with **am** or **pm**; for example, **14:30** or **"2:30 pm"**.

You must enclose the entire entry in double quotes if it contains spaces; for example, **"2:30 pm"** or **"sun 14:30"**.

You can also set a restart time as **never** or **now**. The setting **never** indicates that the BOS Server does not perform the indicated type of restart. For a restart time, the setting **now** is equivalent to specifying the current day and time.

CAUTION:
Never edit the restart times in the BosConfig file directly; use the bos setrestart command only. If you edit the restart times directly, the BOS Server does not recognize the new times until it is restarted and again reads the BosConfig file.

Checking the Current Restart Times

To check the current time settings, issue the **bos getrestart** command:

```
$ bos getrestart -server machine
```

Following is an example of the output from this command:

```
$ bos getrestart /.../abc.com/hosts/fs3

Server /.../abc.com/hosts/fs3 restarts at sun 4:00 am
Server /.../abc.com/hosts/fs3 restarts for new binaries at 5:00 am
```

Setting the General Restart Time

To set the general restart time, perform the following steps:

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.

You can set the general restart time to **never** if you do not want the BOS server to perform a general restart. To set the general restart time to never, enter the following:

```
$ bos setrestart -server machine -general never
```

Setting the New Binary Restart Time

Important Note to Users

OS/390 DFS does not support binary restart time.

Rebooting a Server Machine

Important Note to Users

In OS/390 DFS, stopping and starting the DFS started task is the equivalent of rebooting a server machine. See Chapter 7, “Starting and Stopping the DFS Server in OS/390 DFS” on page 117 for instructions on stopping and starting OS/390 DFS.

Rebooting a Non-OS/390 Server Machine from a Remote Machine Console

1. **To reboot from the console of a remote machine** open a remote connection to the machine you want to reboot (using **telnet** or an appropriate program). To reboot from the local console of the machine you want to reboot, omit this step.
2. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine to be rebooted. If necessary, issue the **bos lsadmin** command to check.
3. Issue the **bos shutdown** command to prepare to power down the machine to be rebooted. This command directs the BOS Server to shut down the other DFS server processes that are running on the machine by changing their status flags in the BOS Server's memory to **NotRun**. The BOS Server does not shut itself down; it terminates safely when you turn off the machine. Include the **-wait** option to be sure that all processes have stopped before performing the next step.

   ```
   $ bos shutdown -server machine -wait
   ```

   The **-wait** option causes the command shell prompt to remain absent until the processes are stopped. If the **-wait** option is omitted, the prompt returns immediately, even if the processes are not yet stopped.
4. Log in as **root** in the native UNIX file system of the machine to be rebooted. For example:

   ```
   $ su root

   Password: root_password
   ```
5. Issue the appropriate reboot command (**/usr/sbin/shutdown** or its equivalent) for the machine to be rebooted. For example:

   ```
   # /usr/sbin/shutdown -Fr
   ```

Chapter 10. Making Filesets and Aggregates Available

Important Note to Users

Some instructions in this chapter describe logging in as **root**. For OS/390 DFS, this should indicate a user defined with **UID = 0**.

In OS/390 DFS, there are two types of non-DCE Local File Systems, the OS/390 Hierarchical File System (HFS) and the Record File System (RFS). An RFS consists of a collection of OS/390 Data Sets (sequential, partitioned, or VSAM) with a common data set name prefix. It can also be a single Partitioned Data Set (PDS or PDSE). The information in this chapter on non-DCE Local File Systems generally refers to both types of non-DCE Local File Systems. For more information on using RFS files from a DFS client, see Chapter 12, “Using OS/390 Data Sets from DFS Clients” on page 235.

In the DCE Local File System, a fileset is defined as a collection of related files that are organized into a single, easily managed unit. Because DCE Local File System filesets are usually smaller in size than standard file systems, and because each DCE Local File System aggregate can house multiple DCE Local File System filesets, DCE Local File System 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 Local File System 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 Local File System file systems (non-DCE Local File System filesets) cannot be replicated in DFS.

This section provides detailed information about how to create, replicate, and back up DCE Local File System filesets, and how to mount DCE Local File System and non-DCE Local File System filesets for use in the DCE namespace. It also explains how to export aggregates and HFS or RFS data sets. (See Chapter 11, “Managing Filesets” on page 197 for information on the tasks involved in the use and maintenance of filesets.)

Notes:

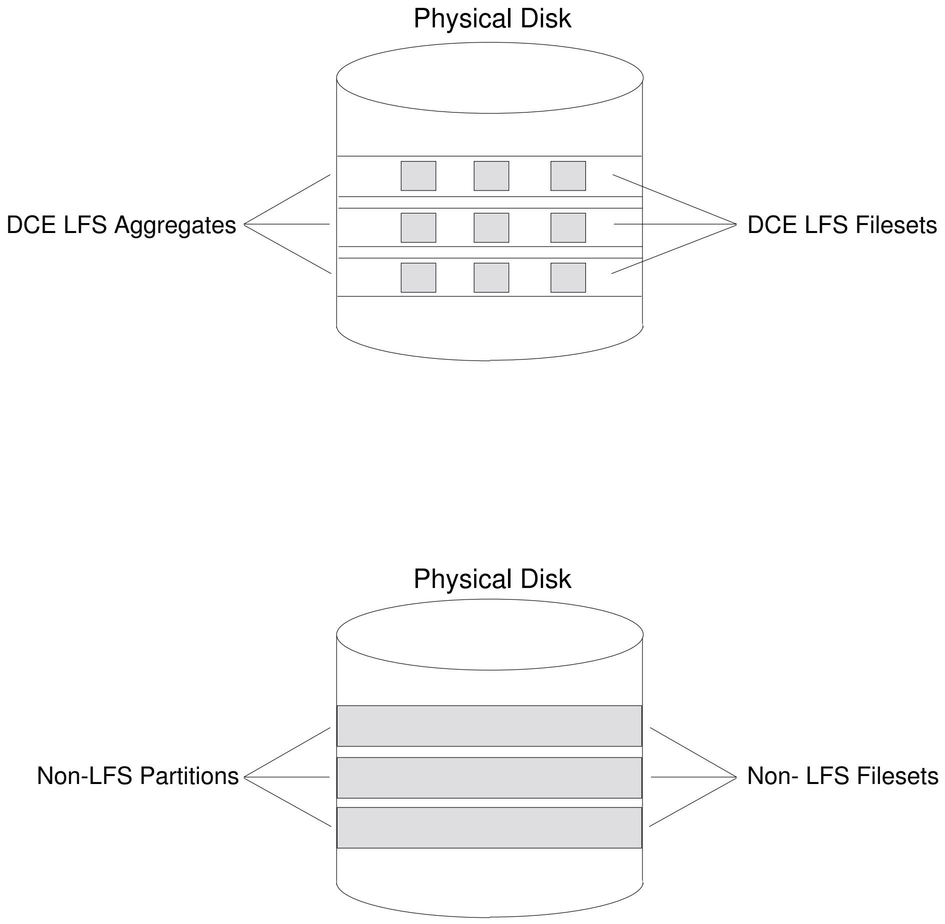
1. This guide uses the term *non-Local File System* to refer to non-DCE Local File System filesets and aggregates. For the OS/390 DFS implementation, non-Local File System and non-DCE Local File System refer to OS/390 Hierarchical File System (HFS) and Record File System (RFS) data sets.
2. The terms *partition* and *disk partition* as used in this chapter may refer to OS/390 Hierarchical File Systems (HFS) when referring to non-Local File System or a ***Logical Volume*** when referring to DCE Local File System.

An Overview of Filesets

A DCE Local File System fileset is a hierarchical grouping of files that is managed as a single unit. A DFS Local File System aggregate on OS/390 is a collection of one or more VSAM Linear Data Sets (LDS) modified to include the DCE Local File System metadata structure that supports DCE Access Control Lists (ACLs), multiple DCE Local File System 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 Local File System, you can export multiple filesets from one aggregate. Because non-Local File System partitions do not support the enhancements that are supported on DCE Local File System aggregates, and because you can store only one fileset on a non-Local File System partition, you can export only one non-Local File System fileset per non-Local File System partition.

Figure 12 illustrates the structural differences between the DCE Local File System and other file systems. The partitioning structure in the DCE Local File System features aggregates, each of which can store multiple DCE Local File System filesets; the partitioning structure in other file systems has partitions that can house only a single non-Local File System file system each.



Disk Partitioning Structures

Figure 12. Comparison of DCE Local File System and non-Local File System

Creating and Using Filesets

Before you can create a DCE Local File System fileset on OS/390, you must first allocate the VSAM Linear Data Set(s) (LDS) that will make up a Logical Volume. After allocating the LDSs, use the **newaggr** command to initialize, or format, the Logical Volume on which the aggregate is to reside. You must then export the aggregate with the **dfsexport** command to make it available in the DCE namespace.

Important Note to Users

A ***Logical Volume*** must be created on OS/390 before **newaggr** can be run. **newaggr** is run as a batch job, as a TSO/E command, or from the OS/390 shell. The *xxx*.SIOESAMP(NEWAGGR) member (where *xxx* is installation dependent) has sample JCL for running this program in batch. This is a two-step job. The first step allocates the VSAM LDSs that make up the ***Logical Volume*** and the second step runs the **newaggr** program to format is as a DCE Local File System aggregate.

newaggr must run as **UID 0** in order to access the **devtab** file.

For information on running **newaggr** from TSO/E, or from the OS/390 shell, refer to “newaggr” on page 737.

After the DCE Local File System aggregate is initialized and exported, a DCE Local File System 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 Local File System 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-Local File System fileset for use on the local machine, you do not use **fts** commands. A disk partition (HFS data set) is equal to one non-Local File System fileset. You may use the ISPF shell, the TSO/E **ALLOCATE** command, or JCL to allocate an HFS data set where a non-Local File System fileset is to reside. The TSO/E **MOUNT** command (or its appropriate equivalent) is then used to mount the partition locally, after which data can be placed on the partition and used locally.

To make a non-Local File System 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-Local File System aggregate* can also be used to refer to an exported partition (HFS file system).

Different Types of DCE Local File System Filesets

There are three types of filesets in the DCE Local File System — read/write, read-only, and backup. Non-Local File System file systems do not have these different types of filesets. When used with DFS, non-Local File System filesets are essentially treated as read/write filesets. However, a partition that houses a non-Local File System 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 Local File System 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 with 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 (with Release Replication) or it can be automated (with 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 14, "Configuring the Backup System" for more information on the backup process).

Figure 13 illustrates the different types of DCE Local File System filesets — read/write, read-only, and backup.

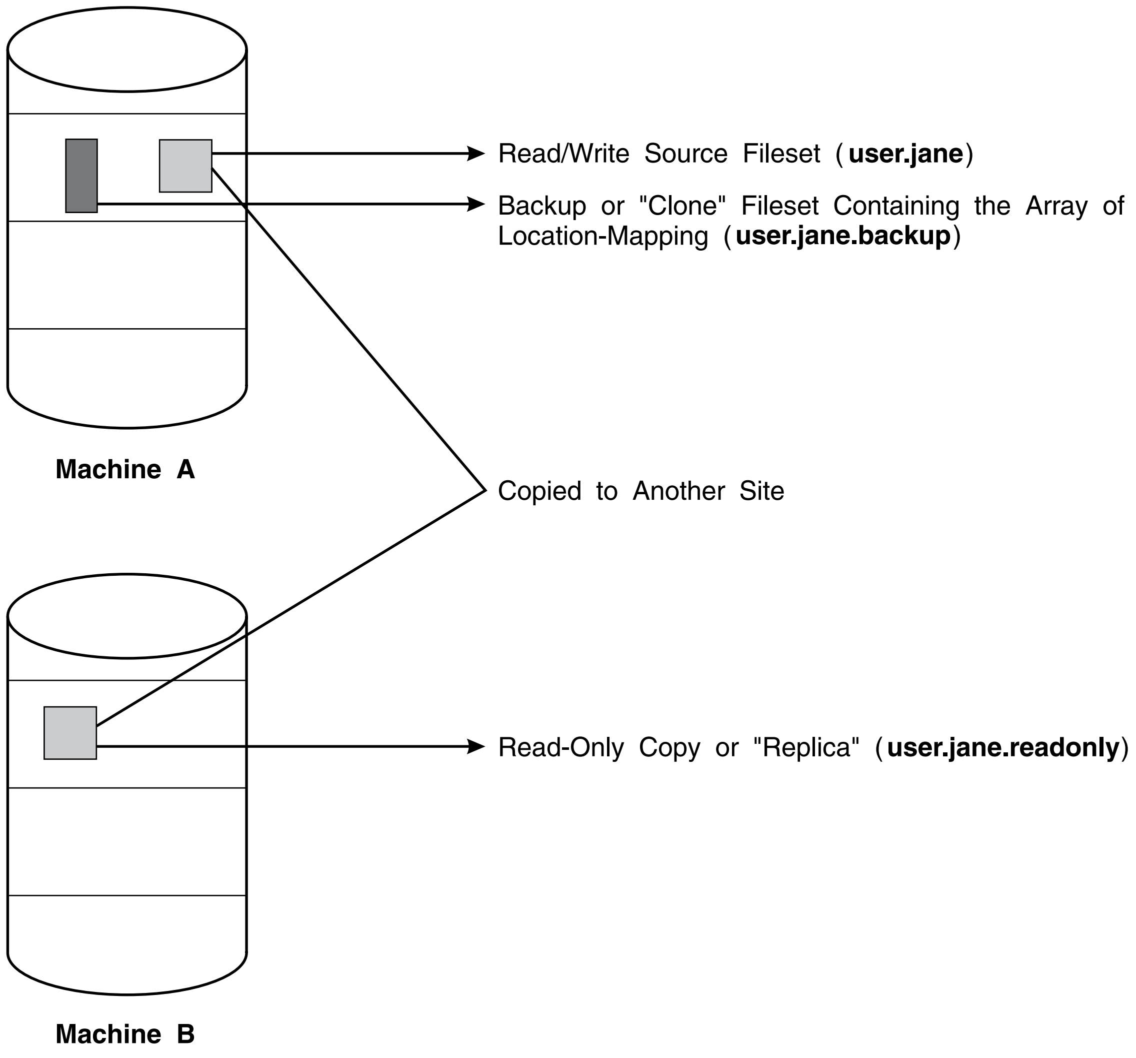

Read/Write Source Fileset (user.jane)
Backup or "Clone" Fileset Containing the Array of Location-Mapping (user.jane.backup)
Machine A
Copied to Another Site
Read-Only Copy or "Replica" (user.jane.readonly)
Machine B

Figure 13. The Different Types of DCE Local File System Filesets

Data Sharing Among the Different Types of DCE Local File System Filesets

When a backup or read-only fileset occupies the same site (File Server machine and aggregate) as its read/write source, DFS attempts to save disk space by having the filesets share data that is the same across the fileset types. 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 that point to the data housed by the read/write source; and
- 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 gives the read/write fileset access to the data without requiring the read/write fileset to 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 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 must acquire additional disk blocks to store changed data, rather than simply pointing to the disk blocks housed by the read-only or backup fileset.

Because of this data-sharing arrangement, DFS provides two statistics on disk usage and limits on disk usage for a fileset. The first statistic, quota, identifies the amount of disk space occupied by all of the files and directories within the read/write fileset including those files and directories whose data is actually housed in another fileset type. The second statistic, allocation, identifies the amount of actual disk space occupied by those files and directories whose data is housed by the fileset.

Users are concerned with the quota statistics only, because it 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 “fts lsquota” on page 669 for more 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 11, “Managing Filesets” on page 197 for information on changing a fileset's quota.)

Administrators are 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 an individual fileset's allocation (and quota) statistics, issue the **fts lsft** command. To check the allocation (and quota) statistics for multiple filesets on a File Server machine, issue the **fts lsheader** command. See Chapter 11, “Managing Filesets” on page 197 for information on the **fts lsft** and **fts lsheader** commands.

Note: There is no way to set or change the allocation for a fileset. It is automatically set during the creation of the fileset and cannot be changed.

Identifying DCE Local File System and Non-Local File System Filesets

Every DCE Local File System and non-Local File System fileset is identified by a unique name and ID number. The following subsections discuss these two forms of fileset identification.

Fileset Names: Every fileset must have a fileset name that is unique within the cell in which it resides. The name is stored in the cell's FLDB. You assign a name to a DCE Local File System 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-Local File System 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 **.** (period)
- The **-** (dash)
- The _ (underscore).

A fileset name must include at least one alphabetic character or an _ (underscore); it cannot consist of just numbers, periods, 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: 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-Local File System fileset names, even though non-Local File System filesets do not need the extensions because they cannot have read-only and backup versions.

With DCE Local File System, 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-Local File System 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.

For filesets that are HFS or RFS filesets, if you use the OS/390 data set name as the fileset name, it can make administration easier in some cases. For example, if you need to use the **fts syncfldb** command because the FLDB has no fileset entry for this fileset (refer to the chapter on deconfiguring DFS in the *OS/390 Distributed File Service DFS Configuring and Getting Started* book), the command will create a fileset entry in the FLDB with the OS/390 data set name as the fileset name.

Fileset ID Numbers: Every fileset also has a fileset ID number that, like a fileset name, is unique within the cell in which the fileset resides. When a fileset is registered in the FLDB with the **fts create** or **fts crfldbentry** command, the FL Server allocates it a fileset ID number, which is stored in the FLDB along with its name. Read/write and backup filesets have their own fileset IDs, which are automatically reserved in the FLDB when the read/write source fileset is registered; all read-only copies of the same read/write fileset share a common fileset ID.

Fileset ID numbers are represented as two positive integers separated by a pair of commas. For example, the ID number of the first fileset in the FLDB is **0,,1**. The integer after the commas is then incremented every time a new fileset is created. When the integer after the commas becomes larger than 2^{32}, the integer before the commas becomes 1 and the integer after the commas returns to 0 (zero).

When specifying a fileset ID in a DFS command, you can omit the integer before the commas if it is a 0 (zero); commands that accept a fileset ID number assume that the first integer is 0 (zero) if it is not supplied. In this case, also omit the two commas. For example, the fileset ID number **0,,1** can be entered as **1**.

Tracking Fileset Locations

The Fileset Location Database (FLDB) is maintained by the FL Server. The FLDB records information about the locations of the filesets in a cell. Users do not have to track the location of a fileset; 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 Local File System 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). 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 (as described in Chapter 14, “Configuring the Backup System”).

Note: A non-Local File System 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-Local File System fileset is stored in the **/opt/dcelocal/var/dfs/dfstab** on the local machine. (For more information on the **dfstab** file see item 5 on page 170 and item 6 on page 173.

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 Local File System 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 (more information about replication parameters appears later in this section); and

- 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 communication 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-Local File System filesets, FLDB entries for non-Local File System filesets do not contain as much information as entries for DCE Local File System filesets do. However, information, such as the ID number and site of the fileset, is recorded in the FLDB. The **fts lsfldb** and **fts lsft** commands display this information.

A Fileset's Header Information: A separate fileset header is stored at each site where a version of a DCE Local File System 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 Local File System 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);
- The storage space used by the fileset, in kilobytes;
- Additional internal information about the state of the fileset and its access status;
- 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 Local File System 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); and
- The day, date, and time that the contents of the fileset were last updated.

Because non-Local File System filesets do not have DCE Local File System fileset headers, only such information as the fileset ID number is available from the machine that houses the fileset. The **fts lsheader** and **fts lsft** commands display this information.

Replicating DCE Local File System Filesets

Replication is the process of creating one or more read-only copies of the read/write version of a DCE Local File System 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 in the event of machine failure. Replication is not available for non-Local File System filesets.

You can manually initiate the replication of a DCE Local File System fileset by using Release Replication, or you can automate the process by using Scheduled Replication. With Release Replication, you issue a command that updates the read-only copies whenever you want to release new read-only copies of the read/write 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 updated. The Replication Server then updates the copies at the specified intervals. This type of replication is useful for filesets whose replication is to be performed asynchronously.

Mounting Filesets

In order to make a DCE Local File System or non-Local File System fileset's contents visible and accessible to users in the DCE namespace, the fileset is attached 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 section all use regular mount points, which are the most common type. (See “Using Mount Points” on page 192 for more information about the other types of mount points.)

Note: An HFS data set is mounted at its root.

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 a mount point. The fileset has a directory structure whose root directory has the same name as the mount point. You can create standard subdirectories within the fileset's root directory. You can also create other mount points there, which appear like subdirectories; these mount points are then associated with files in their own filesets rather than with files in the mount-level directory's fileset.

When the Cache Manager traverses a pathname to locate a file that resides in your cell, it begins at the cell's top-level fileset (**root.dfs**). As it traverses the file's pathname, the Cache Manager accesses a different fileset whenever it encounters a mount point.

Most filesets are mounted with regular mount points. When the Cache Manager encounters a regular mount point in a read-only fileset, it attempts to access the read-only version of the fileset named by the mount point. It also attempts to access the read-only version of any fileset whose mount point it encounters further in the file's pathname. However, if a fileset in the pathname is not replicated, the Cache Manager accesses the read/write version of the fileset. From that point on, it continues to access the read/write version of each fileset it encounters in the remainder of the pathname unless it is explicitly directed to access the read-only (or backup) version of a fileset.

Given how the Cache Manager traverses mount points, to be able to access the read-only version of a fileset, you must replicate all filesets mounted at higher levels in the file system hierarchy. In other words, you must create read-only copies of the fileset that contains the mount point for the fileset and all filesets above it in the file system. (See “Using Mount Points” on page 192 for more about mount points and how they are accessed.)

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 section. If an option or argument is not described with a command in the text, a description of it appears here. See Chapter 19, "Distributed File Service Commands" on page 401 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 hostname (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/lfs1** or **/dev/ufs2**), 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 **/opt/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 **fts** 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 6, "Using Administrative Lists and Keytab Files" on page 103 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.
- The **-localauth** option directs **fts** to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server** (for example, **/.../abc.com/hosts/fs1/dfs-server**). Do not confuse a machine's DFS server principal with its unique **self** identity. (See "Exporting Aggregates and Partitions" for information about DFS server principals.)

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

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

Exporting Aggregates and Partitions

A DCE Local File System aggregate on OS/390 is made up of one or more formatted linear data sets (LDS). The LDSs for each aggregate are specified in **/opt/dcelocal/var/dfs/devtab** and make up the logical volume. The logical volume consists of one or more LDSs which are formatted to create a single aggregate which will later be exported by **dfskern**. (Note, there is a sample **devtab** file in the **/opt/dfsglobal/examples**.) To define a logical volume, edit the file **/opt/dcelocal/var/dfs/devtab** and add the following entry:

```
define_lfs n
```

where *n* is the minor number of the device you are defining. A minor number can be any integer greater than zero. Each logical volume should have a unique minor number. The minor number becomes part of the aggregate's device name.

Following this entry, specify each linear data set you want to include in this logical volume. The following is an example of a complete entry in the **devtab** file for a specific logical volume.

```
* Devtab - Example Entry for a logical volume
define_lfs 1
DFS.DCELFS.AGGR001.LDS00001
DFS.DCELFS.AGGR001.LDS00002
DFS.DCELFS.AGGR001.LDS00003
```

Lines beginning with an asterisk are comments. The next line defines the type of file system (lfs) and the minor device number, 1 in the example. The three lines following the definition of the logical volume are the names of the LDSs that make up the DCE Local File System aggregate.

The logical volume defined is known as **/dev/lfs1**, with 1 being the device's minor number. It is recommended that the minor device's number be specified without leading zeroes. This logical volume name should be used when specifying the device in DCE Local File System utilities such as **newaggr** and in the **dfstab** file for exporting to **dfskern**.

An HFS (Hierarchical File System) aggregate is a single HFS file and is a non-Local File System aggregate. The whole aggregate is also a single complete fileset to the DFS server. The HFS file must be mounted to the OS/390 UNIX address space before exporting it. Allocating and mounting an HFS file system is described in the *OS/390 UNIX System Services Planning* book, SC28-1890.

You must add an entry in **/opt/dcelocal/var/dfs/devtab** which maps a minor device number to the HFS file system you wish to export. For each HFS file system add the following entries:

```
* HFS devices
define_ufs 2
omvs.u.abc
```

An RFS (Record File System) aggregate is an OS/390 data set name that is either the name of a partitioned data set or is the prefix of a set of data sets and is a non-Local File System aggregate. The whole aggregate is also a single complete fileset to the DFS server.

You must add an entry in **/opt/dcelocal/var/dfs/devtab** which maps a minor device number to the RFS fileset you wish to export. For each RFS fileset add the following entries:

```
* RFS devices
define_ufs 3 rfs
USERA
```

In the previous two examples, lines beginning with an asterisk are comments. The next line defines the type of file system (**ufs**), the minor device number, and whether the **ufs** file system is an **hfs** (hierarchical file system) subtype or an **rfs** (record file system) subtype. The default subtype is **hfs**. In this case, we are defining a **ufs** with a subtype of **rfs**.

Note: From now on, we will generally refer to this as an **rfs** fileset. You should keep in mind, however, that it is really a **ufs** fileset with a subtype of **rfs**. When we make comments about a **ufs** fileset, they pertain to both subtypes - **hfs** and **rfs**.

As with a DCE Local File System aggregate, the HFS fileset and the RFS fileset have a minor number describing the device, **2** in the HFS example and **3** in the RFS example. This number can be any integer greater than zero. Each **ufs** fileset must have a unique minor device number. (That is, a **ufs** fileset can have the same minor device number as an **lfs** fileset, but each **ufs** fileset, whether it is an **hfs** subtype or an **rfs** subtype, must have a unique minor device number.) Similarly, two DCE Local File System aggregates cannot have the same minor device number.

The third line in the example is the name of the HFS file system or RFS fileset (prefix) you wish to export. The third line also supports an optional parameter in addition to the data set name in the **devtab** for HFS or RFS filesets to control character data translation. The possible values for the translation control parameter are the following:

binary	Do not translate the data.
text	Translate the data with the default translation tables. The default for local data is the local code page for the process. The code page for "wire" data is ISO 8859-1.

The following example uses the **text** parameter.

```
* HFS devices
define_ufs 2
omvs.u.abc text
```

An additional translation control parameter value is available for HFS filesets and an additional translation configuration file is available for HFS. The additional translation control parameter value for HFS is:

auto	Determine whether to translate the data based on the contents of the data. (See the **Usage** section in "devtab" on page 368 for a full description on how the determination is made.)

The additional translation configuration file for HFS is the **hfsattr** file. It allows you to specify that the decision to translate should be based on the file name extension (suffix). The pathname of the **hfsattr** file is specified in the DFSKERN **_IOE_HFS_ATTRIBUTES_FILE** environment variable. If this environment variable is specified, the DFS server attempts to use the file name extension (suffix) for HFS files to determine if the data should be translated.

If the translation control parameter is omitted the default is controlled by the **_IOE_HFS_TRANSLATION** environment variable setting (for HFS filesets) and the **_IOE_RFS_TRANSLATION** environment variable setting (for RFS filesets) in the **dfskern** process. See Appendix B, "Sharing Data Between OS/390 UNIX Applications, Commands, and DFS Clients" on page 795 for more information.

Finally, the **attrfile** *attributes_file* parameter (where *attributes_file* is the name of the attributes file that controls the data set creation, processing, and site attributes for this RFS fileset) is supported on the same line as the data set name in the **devtab** for RFS filesets. The default attributes file for RFS filesets is specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable specified in the **dfskern** process. For a complete list of options for **devtab** see "devtab" on page 368.

The logical volume defined is known as **/dev/ufs2**, with **2** being the minor device number. This logical volume name should be used when specifying the device in the **dfstab** file for exporting to **dfskern**.

Before exporting a DCE Local File System aggregate or a non-Local File System partition (non-Local File System aggregate) from a File Server machine, you must ensure that an RPC binding exists for the DCE pathname of the machine and that a corresponding DFS server principal exists 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 registered in the Registry Database, and the server entry is registered in the FLDB. (These prerequisites and additional requirements for exporting are described further in the following section.)

Prior to exporting a DCE Local File System aggregate, you must use the **newaggr** command to construct the aggregate from a pre-allocated Logical Volume consisting of one or more VSAM Linear Data Sets. You run **newaggr** once for each aggregate that you want to create.

The Logical Volume to be initialized as a DCE Local File System aggregate must be neither mounted locally nor exported to the DCE namespace when you issue the **newaggr** command. Conversely, before exporting a non-Local File System partition for use as an aggregate in DFS, you must create the HFS file system and mount it locally.

Note: You will also need a **devtab** entry for the Cache Manager aggregate, see Chapter 13, “Configuring the Cache Manager” on page 251 for more information.

To make data on a DCE Local File System aggregate or non-Local File System 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 **/opt/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. The **dfsexport** command will not export an aggregate or partition that is already exported. 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.

Important Note to Users

For OS/390 DFS, **dfsexport** is set up to run automatically after **dfskern** (fxd) has successfully initialized during the DFS address space startup. For more information, see “Starting DFS Server Daemons During System IPL” on page 123.

Because a non-Local File System 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 this 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 the partition contains. The **fts crmount** command makes a fileset visible in the DCE namespace.

Conversely, the **dfsexport** command must be used to export a DCE Local File System aggregate to the DCE namespace *before* the aggregate can store filesets. Once a DCE Local File System aggregate is exported, filesets can be created on it with the **fts create** command, and mount points can be created for the filesets with the **fts crmount** command. You specify a name for a DCE Local File System fileset with the **fts create** command; the fileset is automatically assigned an ID number and registered in the FLDB.

The following sections describe these steps and the commands that are used to perform them in more detail. See Chapter 19, “Distributed File Service Commands” on page 401 for more information on a specific command.

Preparing for Exporting

Several prerequisites must be met before you can export either a DCE Local File System aggregate or a non-Local File System 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).
- A Security Server process (**secd**) must be running in the cell, and a security client process must be running on the machine (use the appropriate Security Service command to verify that the processes are running).

- The **dced** process must be running on the machine; use the appropriate **dcecp** command to verify that the process is running.
- An FL Server process must be running in the cell. You can use the **bos status** command to verify that the **flserver** process is running or you can use the **fts lsfldb** command to list information about the **root.dfs** fileset, which must exist, thus verifying that the **flserver** is actually functioning.

In addition, an RPC binding must exist for the DCE pathname of the File Server machine in CDS, a corresponding DFS server principal must exist for the machine in the Registry Database, and a server entry must exist for the machine in the FLDB. The following subsections describe these topics and additional preparation that must be completed if the machine is to export aggregates or partitions.

Creating an RPC Binding for a File Server Machine: 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**).

Notes:

1. 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.
2. 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 the CDS must have a name of the form **/.../***cellname***/hosts/***hostname***/self**.

Creating a DFS Server Principal for a File Server Machine: A File Server machine must also have a DFS server principal registered in the local Registry Database. The DFS server principal name is used to establish an authenticated connection to the DFS server machine.

In the 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 6, "Using Administrative Lists and Keytab Files" for more information on server encryption keys.)

Use the **dcecp principal create** command to create a DFS server principal and associated account in the Registry Database for the File Server machine from which aggregates and partitions are to be exported. The DFS server principal must be of the form **/.../***cellname***/hosts/***hostname***/dfs-server**.

Creating a Server Entry for a File Server Machine: 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.

To create a server entry for a File Server machine, perform the following steps:

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, hostname, 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. 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** *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. 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. If this option is omitted, the default is 0 (zero), meaning that an unlimited number of FLDB 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*). 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. 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.

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 server 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.

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, hostname, or IP address of the server machine whose entry in the FLDB is to be displayed. Use this option or use the **-all** option.
- The **-all** option specifies that the entries for all server machines in the FLDB are to be displayed. Use this option or use the **-server** option.

Editing a Server Entry for a Machine: To edit the server entry for a File Server machine, perform the following steps:

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, hostname, 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 appends 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 FLDB 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 one exists. 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 (displayed as `<nil>`) replaces the group that currently owns the server entry; the entry is unchanged in this regard if no group presently owns the server entry. Use this option or use the **-owner** option; omit both options to leave the current owning group unchanged.

Deleting a Server Entry for a Machine: To remove the server entry for a File Server machine, perform the following steps:

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, hostname, 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.

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 Local File System filesets.

Important Note to Users

In OS/390 DFS, the function of the **dfsbind** process that supports DFS servers is incorporated into the **fxd** process. The **fxd** is run as part of the **dfskern** process.

The following section 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. On OS/390, this means as an OS/390 UNIX UID 0 user.
2. Start the DFS address space using the **-nodfs** parameter.

   ```
   start dfs,parm='-nodfs'
   ```

3. Start the BOS Server (**bosserver** process) on the machine using the **-noauth** option to disable DFS authorization checking on the server machine. (See Chapter 6, “Using Administrative Lists and Keytab Files” for a thorough description of DFS authorization checking.) The process automatically creates the **admin.bos** file when it starts.

```
modify dfs,start boserver -noauth
```

The **-noauth** option starts the **bosserver** with DFS authorization checking turned off.

4. 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 **bos** to use the unprivileged identity **nobody** as the identity of the issuer.

5. Add a server encryption key to the keytab file on the machine with the **bos genkey** command, again using the **-noauth** option. (See Chapter 6, “Using Administrative Lists and Keytab Files” on page 103 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 **bos** to use the unprivileged identity **nobody** as the identity of the issuer.

6. 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 **bos** to use the unprivileged identity **nobody** as the identity of the issuer.

7. If you plan to export HFS or RFS data sets, you must create the identity mapping file. See “Creating the Identity Mapping Input File” on page 174 for instructions on creating the identity mapping file. If you do not plan to export HFS or RFS data sets, do not create this file. Continue with step 9.

8. If you plan to export RFS data sets, you should create an attributes file. This controls how data sets are created and processed by DFS clients. See “Attributes File (rfstab)” on page 341 for more information.

9. Start the **dfskern** process to initialize the File Exporter and the DFS kernel.

   ```
   modify dfs,start dfskern
   ```

   Note: During **dfskern** initialization a small file (**opt/dfslocal/var/dfs/rfsfile**) is created (or updated) to keep track of RFS file identifiers. This file should not be changed or erased.

10. Log out as **root** from the machine to return to your authenticated DCE identity.

11. Start the Fileset Server (**ftserver** process) with the **bos create** command. (See Chapter 9, “Monitoring and Controlling Server Processes” 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 "ftserver envar('_EUV_HOME=/opt/dfslocal/home/ftserver') / >dd: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 hostname 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** option provides the name of the fileset server load module, an environment variable specifying a home directory for the ftserver, and a redirection indicator for the ftserver's **STDOUT**.
    - The slash (**/**) before **>dd:ftserver** separates the C runtime parameters passed to the program.

12. 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 hostname 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.

13. 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 "rpserver envar('_EUV_HOME=/opt/dfslocal/home/repserver') / >dd:rpserver"
    ```

    - 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 hostname 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** option provides the name of the replication server load module, an environment variable specifying a home directory for the repserver, and a redirection indicator for the repserver's **STDOUT**.
- The slash (**/**) before **>dd:ftserver** separates the C runtime parameters passed to the program.

14. After the Fileset Server process is started, use the **fts statftserver** command to verify that the process is performing requested actions. This command is useful mainly if you believe the process is not functioning properly.

    ```
    $ fts statftserver -server machine
    ```

    The **fts statftserver** command displays the message `No active transactions on` *`machine`* if the Fileset Server is functioning properly. It displays additional information if the Fileset Server is currently performing an action. Depending on the information displayed, the Fileset Server may or may not be functioning properly.

Exporting DCE Local File System Aggregates

The following sections introduce and describe the steps involved in initializing and exporting a DCE Local File System aggregate. Before exporting a DCE Local File System 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, as described in “Preparing for Exporting” on page 159.
- An RPC binding must exist for the DCE pathname of the machine, as described in “Creating an RPC Binding for a File Server Machine” on page 160.
- A DFS server principal must exist for the machine. See “Creating a DFS Server Principal for a File Server Machine” on page 160 for instructions on this.
- A server entry must exist for the machine, as described in “Creating a Server Entry for a File Server Machine” on page 161.
- The steps under “Preparing for Exporting” on page 159 beginning with step 1 on page 163 through step 10 on page 164.

You must also initialize the aggregate by formatting it with the **newaggr** command before it can be exported.

An Overview of Initializing DCE Local File System Aggregates

Important Note to Users

A ***Logical Volume*** must be created on OS/390 before **newaggr** can be run. **newaggr** is run as a batch job, as a TSO/E command, or from the OS/390 shell. The *xxx*.SIOESAMP(NEWAGGR) member (where *xxx* is installation dependent) has sample JCL for running this program in batch. This is a two-step job. The first step allocates the VSAM LDSs that make up the ***Logical Volume*** and the second step runs the **newaggr** program to format it as a DCE Local File System aggregate.

In OS/390 DFS, you must do the following before **newaggr** can be run:

- Create a Logical Volume on OS/390.
- An entry must be added to the **devtab** file.
- An entry must be added to the **dfstab** file.

Prior to creating a DCE Local File System aggregate, use the **newaggr** command to initialize the Logical Volume on which the aggregate is to reside by formatting it for use as a DCE Local File System aggregate. The **newaggr** command creates the metadata structure used by the DCE Local File System for ACL support, logging, multiple fileset storage, and other fileset-related operations. It also allocates temporary space for use by the DCE Local File System log for faster restarts after system failures. The DCE Local File System 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 Logical Volume being initialized, the Logical Volume 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. (Note that a non-Local File System 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: DCE Local File System reserves a variable amount of disk space on every DCE Local File System aggregate. By default, DCE Local File System 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 Local File System 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.

Initializing a DCE Local File System Aggregate: To initialize a DCE Local File System aggregate, perform the following steps:

Attention: Do not use the **newaggr** command to initialize a non-Local File System partition. Also, do not use the command on a partition or aggregate that is currently exported to the DCE namespace; the command fails in these cases.

1. Modify the **devtab** file. The **devtab** file describes all the Linear Data Sets that make up a Logical Volume. It consists of the names of the Linear Data Sets that make up the Logical Volume.

   ```
   * Devtab - Example Entry for a logical volume
   define_lfs 1
   DFS.DCELFS.AGGR001.LDS00001
   DFS.DCELFS.AGGR001.LDS00002
   ```

2. Modify the **dfstab** file. The **dfstab** file specifies DCE Local File System aggregates and non-Local File System partitions that can be exported.

```
* Dfstab - Example Entry
/dev/lfs1   lfs1   lfs  1
```

3. Use JCL similar to the following to allocate one or more VSAM Linear Data Sets that will make up a Logical Volume. The Linear Data Sets may reside on the same or different OS/390 disk volumes. The size of the Linear Data Sets determines the size of the DCE Local File System.

```
//DFSDEFIN JOB ,'DFS Define LDSs',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*-----------------------------------------------------------------
//*
//*  Allocate a VSAM Linear Datasets for use as a Logical Volume
//*
//*-----------------------------------------------------------------
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=PH2020
//DASD1    DD     DISP=OLD,UNIT=3390,VOL=SER=PH2021
//SYSIN    DD     *
  DEFINE CLUSTER (NAME(DFS.DCELFS.AGGR001.LDS00001) VOLUMES(PH2020) -
    LINEAR CYL(100 0) SHAREOPTIONS(2) )
  DEFINE CLUSTER (NAME(DFS.DCELFS.AGGR001.LDS00002) VOLUMES(PH2021) -
    LINEAR CYL(100 0) SHAREOPTIONS(2) )
//
```

4. Run the **newaggr** JCL to initialize the Logical Volume as a DCE Local File System aggregate:

```
//DFSNEWAG JOB ,'DFS NewAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*-----------------------------------------------------------------
//*
//*  Allocate a VSAM Linear Dataset(s) for use as a Logical Volume
//*
//*  This version of newaggr also "loads" the VSAM linear data
//*  set(s) if they have not been previously loaded.
//*
//*  NOTES:
//*        - Note that the "/dev/lvm" *must* be lower case!
//*        - Note that the extra '/' is required because LE
//*          runtime parameters are separated from program
//*          parameters by a '/'.
//*
//*  The syntax of the newaggr command is as follows:
//*
//*  newaggr -aggregate name -blocksize bytes -fragsize bytes
//*        [-initialempty blocks] [-aggrsize blocks]
//*        [-logsize blocks] [-overwrite] [-verbose] [-noaction]
//*
//*-----------------------------------------------------------------
//NEWAGGR  EXEC   PGM=IOENEWAG,
// PARM=('//dev/lfs1 8192 1024 -verbose')
//*
//SYSPRINT  DD    SYSOUT=*
//SYSUDUMP  DD    SYSOUT=*
//*
```

- The **-aggregate** *name* option in MVS is the device name or the aggregate name for the Logical Volume to be initialized as a DCE Local File System aggregate. The device name and aggregate name are defined by the **dfstab** entry. For example, **/dev/lfs1**.
- The **-blocksize** *bytes* option is the number of bytes to be available in each DCE Local File System block on the aggregate. Allowable values are the powers of 2 from 4096 to 65,536.
- The **-fragsize** *bytes* option is the number of bytes to be available in each DCE Local File System 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 Local File System 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 Local File System 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 Local File System blocks to be reserved for the log on the aggregate. This value cannot exceed the number of DCE Local File System 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 Local File System blocks on the aggregate (**-aggrsize**) is used.
- The **-overwrite** option specifies that an existing file system found on the partition can be overwritten. If this option is omitted and a file system is found on the partition, the command informs you that a file system already exists. It then terminates with an exit code of at least 16 without overwriting the existing file system.
- The **-noaction** option directs the command to display information about what it would do without actually modifying the partition. Include the other options as you would to actually execute the command. The command displays the default values it would use for its options and informs you if the partition already contains a file system.

Exporting a DCE Local File System Aggregate: To export a DCE Local File System aggregate, perform the following steps:

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. You must also 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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) 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. (You need to have the **w** and **x** permissions for any directories in non-Local File System filesets.)
3. Log in as **root** on the machine from which the aggregate is to be exported. On OS/390, this means as an OS/390 UNIX UID 0 user.
4. You must modify the **devtab** file as described in “Exporting Aggregates and Partitions” on page 156.

5. Use a text editor to edit the **dfstab** file to include an entry for the DCE Local File System 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 device name of the aggregate (for example, **/dev/lfs1**). For additional information regarding device names, see "Exporting Aggregates and Partitions" on page 156.
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**). For additional information regarding aggregate names, see "Exporting Aggregates and Partitions" on page 156.
File System Type	The identifier for the file system type of the aggregate. For DCE Local File System 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.)

   For example, the following entry from a **dfstab** file is for a DCE Local File System aggregate:

   ```
   /dev/lfs1 lfs1 lfs 1
   ```

6. 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 Local File System aggregates or only non-Local File System partitions. Use either this option or 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 either the **-aggregate** or **-all** option.
   - The **-type** *name* option is the file system type to be exported. Specify **lfs** to export only DCE Local File System aggregates; specify **ufs** to export only non-Local File System 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.
   - On OS/390 you can export the aggregates using the OS/390 **MODIFY** command. For example:

     ```
     modify dfs,start export
     ```

7. Log out as **root** from the machine to return to your authenticated DCE identity.

8. Issue the **fts create** command to create a DCE Local File System fileset on the aggregate and register the fileset in the FLDB. The FL Server allocates a unique fileset ID number for the fileset. (See "Creating Read/Write DCE Local File System Filesets" on page 179 for detailed information about DCE Local File System fileset creation.)

   ```
   $ fts create -ftname name -server machine -aggregate name
   ```

   The **-ftname** *name* option is the complete name to be associated with the fileset being created. The name can contain no more than 102 characters and it must contain at least one alphabetic character or an _ (underscore). (See “Fileset Names” on page 152 for more information on fileset naming conventions.)

Repeat the **fts create** command for each fileset you want to create on the aggregate.

9. Enter the **fts crmount** command to create a mount point in the file system for the new DCE Local File System fileset. This makes the contents of the fileset visible to other users. (See “Using Mount Points” on page 192 for more information about mounting filesets.)

   ```
   $ fts crmount -dir directory_name -fileset {name | ID}
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

   Repeat the **fts crmount** command for each fileset created in the previous step.

Exporting Non-Local File System Partitions

Important Note to Users

- This guide uses the term *non-Local File System* to refer to non-DCE Local File System filesets and aggregates. In the OS/390 DFS implementation, non-Local File System and non-DCE Local File System refer to OS/390 Hierarchical File Systems (HFS) and Record File Systems (RFS).
- The terms *partition* and *disk partition* as used in this chapter may refer to OS/390 Hierarchical File Systems (HFS) when referring to non-Local File System or a Logical Volume when referring to DCE Local File System.
- This section has several references to “**newfs** and **fstab** (or their equivalents)” concerning Non-Local File System Partitions (file systems). These are not supported on OS/390 DFS.
- In order to map incoming principal DCE identities to OS/390 identities, you must configure the MAP ID file before starting DFS or you can use the RACF identity mapping function. The DFSKERN environment variable **_IOE_MVS_IDMAP_SAF** controls which facility is used.

This section describes the steps that are involved in exporting a non-Local File System partition; after it is exported, a non-Local File System partition can be referred to as a non-Local File System aggregate. Before exporting a non-Local File System partition to the DCE namespace, the prerequisites described in “Preparing for Exporting” on page 159 must be met:

- The necessary CDS, Security Service, RPC, and DFS server processes must be running. See “Preparing for Exporting” on page 159.
- An RPC binding must exist for the DCE pathname of the machine. See “Creating an RPC Binding for a File Server Machine” on page 160.
- A DFS server principal must exist for the machine. See “Creating a DFS Server Principal for a File Server Machine” on page 160.
- A server entry must exist for the machine.
- Perform the instructions in step 1 on page 163 through step 6 on page 164 in “Preparing a File Server Machine for Exporting” on page 163.
- An identity mapping file must exist on the machine or the RACF identity mapping function may be used. The DFSKERN environment variable **_IOE_MVS_IDMAP_SAF** controls which facility is used. See “Mapping DCE User IDs to OS/390 User IDs” on page 174 for more information on mapping DCE user IDs to OS/390 user IDs.
- The **devtab** file must have an entry for your non-Local File System partition to be exported.

You must also have created and locally mounted the partition. Refer to the *OS/390 UNIX System Services Planning* book, SC28-1890, for more information on how to create and mount OS/390 HFS file systems.

Note: For OS/390 DFS, non-Local File System partitions can refer to OS/390 Hierarchical File Systems (HFS). There should not be any local file system activity occurring on an HFS file system before it is exported to the DFS filespace. Otherwise, an inconsistent DFS token state can occur. When you configure the DFS File Server on OS/390 (DFSKERN) to export HFS file systems, you must start DFS at system start time to ensure that the DFS token state is correct. See *OS/390 Distributed File Service DFS Configuring and Getting Started* for more information.

To export OS/390 Data Sets (referred to as Record File Systems (RFS)), local mounting is not required or supported.

To export a non-Local File System partition, perform the following steps:

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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) 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. (You need to have the **w** and **x** permissions if the directory is in a non-Local File System fileset.)
3. Provide a name for the fileset (non-Local File System 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-Local File System 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-Local File System fileset cannot have these versions. Use the read/write ID number returned by the command as the fileset ID in the **dfstab** file.

   ```
   $ fts crfldbentry -ftname name -server machine -aggrid ID
   ```

   - The **-ftname** *name* option is the complete name to be associated with the fileset being registered. The name can contain no more than 102 characters, and it must contain at least one alphabetic character or an _ (underscore). (See “Fileset Names” on page 152 for more information on fileset naming conventions.)

     If you use the OS/390 data set name as the fileset name, it can make administration easier in some cases. For example, if you need to use the **fts syncfldb** command because the FLDB has no fileset entry for this fileset (refer to the chapter on deconfiguring DFS in the *OS/390 Distributed File Service DFS Configuring and Getting Started* book, for information on renaming a machine's DCE host name), the command creates a fileset entry in the FLDB with the OS/390 data set name as the fileset name.
   - 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. On OS/390, this means as an OS/390 UNIX UID 0 user.
5. You must modify the **devtab** file as described in “Exporting Aggregates and Partitions” on page 156.

6. Use a text editor to edit the **dfstab** file to include an entry for the non-Local File System 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. (Since a non-Local File System 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 device name of the partition. For example, **/dev/ufs1**.
Aggregate Name	The name to be associated with the exported partition. The aggregate name of a non-Local File System partition can be 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-Local File System 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-Local File System partition:

```
/dev/ufs1  hfs1  ufs  101 0,,10
```

7. 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 Local File System aggregates or only non-Local File System partitions. Use this option or 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 either the **-aggregate** or **-all** option.
- The **-type** *name* option is the file system type to be exported. Specify **lfs** to export only DCE Local File System aggregates; specify **ufs** to export only non-Local File System 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.
- On OS/390 you can export the aggregates using the OS/390 **MODIFY** command. For example:

  ```
  modify dfs,start export
  ```

8. Log out as **root** from the machine to return to your authenticated DCE identity.
9. Enter the **fts crmount** command to create a mount point in the file system for the new non-Local File System 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.

Note: If the parent directory resides in an OS/390 fileset, it must be contained in a DCE Local File System Aggregate or an HFS Aggregate. (That is, you cannot create a DFS mount point in an RFS fileset.)

10. When exporting RFS filesets, the DFS server OS/390 user ID (for example, DFS1) must have RACF ALTER authority to every file in each RFS fileset that is being exported. For example,

    ```
    PERMIT 'USERA.**' USERID(DFS1) ACCESS(ALTER)
    ```

    Alternatively, you may give the DFS server OS/390 user ID the OPERATIONS attribute. For example,

    ```
    ALTERUSER DFS1 OPERATIONS
    ```

Mapping DCE User IDs to OS/390 User IDs

The DFS server enables DFS clients to access non-DCE Local File System filesets (both HFS and RFS).

The local security subsystem determines if the client is authorized to use the resource. Because the local security subsystem is not integrated with the DCE Security service, the local security subsystem cannot recognize DCE user IDs.

The administrator has to establish a relationship (that is, a **mapping**) between a user's DCE user ID and OS/390 user ID. This section describes two methods that OS/390 DFS may use to map DCE user IDs to OS/390 user IDs. The methods include:

- Identity mapping file (page 174), or
- Resource Access Control Facility (RACF) identity mapping function (page 178).

Note: Although RACF is mentioned, any OS/390 external security manager (ESM) that has equivalent support can be used instead of RACF.

In some cases, the DFS client request is denied if the DFS server cannot determine a mapping to an OS/390 user ID. For example, if the user is unauthenticated or not mapped or mapped but not in RACF and if **_IOE_MVS_DFSDFLT** is not specified or the **_IOE_MVS_DFSDFLT** user ID is not in RACF the mapping function denies the request.

Using the Identity Mapping File Method:

If you do not use the RACF Identity Mapping Function method then you can use this method.

Mapping the DCE user ID to the OS/390 user ID is a four-fold procedure:

1. Creating the **Identity Mapping Input File**.
2. Creating the **Identity Mapping File**.
3. Setting the **_IOE_MVS_IDMAP** environment variable.
4. Ensuring the **_IOE_MVS_IDMAP_SAF** environment variable is not specified or is set to **NO** in the **dfskern** process.

Each of these procedures is described in the following sections.

Creating the Identity Mapping Input File: The Identity Mapping input file is a text file that the administrator creates and maintains. This must be created as an HFS file.

Note: The DFS server administrator must also map his DCE user ID to his OS/390 user ID.

The Identity Mapping input file contains one or more identity mapping declarations and has the following general format:

DCE-user-ID1
OS/390-user-ID1
DFS-server-name

DCE-user-ID2
OS/390-user-ID2
...

This format illustrates the two types of entries that can exist on the Identity Mapping input file. The first entry has three elements: DCE user ID, OS/390 user ID, and DFS server name. The second entry only has two elements: DCE user ID and OS/390 user ID. These are explained later in this chapter. A blank line is required between entries. The following list explains each element in an identity mapping entry:

DCE-user-ID	Is the client's DFS identity. This may either be a simple DCE principal name (for clients within the cell) or a fully qualified global name (for clients within and outside the cell). The DCE user ID can be up to 256 characters in length.
OS/390-user-ID	Is the OS/390 user ID of the client. All potential DFS server clients must have user IDs on the OS/390 host where the DFS server is running.
DFS-server-name	Is the name of the CDS object that represents a specific DFS server. This is optional. If the mapping entry includes the name of a DFS server, the mapping is valid only for that DFS server. If the entry does not include the name of a DFS server, the mapping is valid for all servers that are running on the host system that use the Identity Mapping input file (for example, Application Support for Information Management System). The *DFS-server-name* for a DFS server is specified in the **_IOE_MVS_SERVER** environment variable in the **dfskern** process.

The first entry type (with three elements) is known as the **specific mapping** to a DFS server. The second entry type (with two elements) is known as the **default mapping** for a DFS server user. You can have one or both of these entry types in your Identity Mapping input file.

Although not required, it is recommended that all servers that require identity mapping on a host system share the same Identity Mapping input file (for example, multiple OS/390 DCE application support servers should share the same Identity Mapping input file). Figure 14 shows a sample Identity Mapping input file.

```
smith
CMSMITH
/.:/subsys/dce/dfs

jones
TSJONES
```

Figure 14. Example Identity Mapping Input File

In the first entry, the DCE user ID **smith** is mapped to the OS/390 user ID **CMSMITH**. This mapping is effective only when accessing the DFS server whose CDS name is **/.:/subsys/dce/dfs**. In the second entry, the DCE user ID **jones** is mapped to the OS/390 user ID **TSJONES**. This mapping is effective when accessing **any** DFS server on the host system.

If a user has both specific mapping and default mapping entries with the same principal in the input file, the specific mapping entry overrides the default mapping entry. Consider an Identity Mapping input file that has the following entries:

```
smith
CMSMITH

smith
TSJOHN
/.:/subsys/dce/dfs
```

The second entry (specific) overrides the first entry.

Each user can only have one default mapping for every DFS principal in the file. For example, the following mapping entries are **not** allowed:

```
smith
CMSMITH

smith
TSJOHN
```

Also, each user can only have one specific mapping entry to a particular DFS server for every DFS principal in this file. For example, the following entries are **not** allowed:

```
smith
CMSMITH
/.:/subsys/dce/dfs

smith
TSJOHN
/.:/subsys/dce/dfs
```

Creating the Identity Mapping File: The **Identity Mapping File** is a binary file that is created by running the **mapid** program on the Identity Mapping input file. The Identity Mapping file is created as an HFS file.

The **mapid** program is run with two parameters as follows:

mapid *input-file output-file*

where:

input-file	Is the HFS pathname of the Identity Mapping input file.
output-file	Is the HFS pathname of the Identity Mapping output file.

Mapid can be run from TSO, the OS/390 shell, or in batch. For example, you can enter the following from TSO:

mapid "/opt/dfslocal/home/dfskern/idmap.i" "/opt/dfslocal/home/dfskern/idmap.o"

Important Note to Users

This section includes information on using the **mapid** command. **mapid** is run as a batch job, as a TSO/E command, or from the OS/390 shell. The *xxx*.SIOESAMP(MAPID) member (where *xxx* is installation dependent) has sample JCL for running this program in batch.

For information on running **mapid** from TSO/E, or from the OS/390 shell, refer to “mapid” on page 735.

Figure 15 on page 177 shows an example JCL that is used to run this program.

```
//*JOBCARD...
//*
//*****************************************************************
//MAPID    EXEC PGM=IOEMAPID,REGION=0M,TIME=1440,
//      PARM=('/DD:INFILE DD:OUTFILE')
//*****************************************************************
//* Parameters
//*****************************************************************
//INFILE   DD PATH='/opt/dfslocal/home/dfskern/idmap.i',
//            PATHMODE=(SIRWXU,SIRGRP,SIXGRP),
//            PATHOPTS=(ORDONLY),PATHDISP=(KEEP)
//OUTFILE  DD PATH='/opt/dfslocal/home/dfskern/idmap.o',
//            PATHMODE=(SIRWXU,SIRGRP,SIXGRP),
//            PATHOPTS=(ORDWR,OCREAT),PATHDISP=(KEEP)
//***************
//**************************************************************
//SYSOUT   DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Figure 15. MAPID Program JCL

Note: The Identity Mapping File must be regenerated every time the Security server host is reconfigured in the cell.

Setting the _IOE_MVS_IDMAP Environment Variable: The **_IOE_MVS_IDMAP** environment variable must be set to the name of the Identity Mapping Output file used by the DFS server. The declaration of this environment variable can be made in the **envar** file of the **dfskern** process located in **/opt/dfslocal/home/dfskern/envar**. (For other ways of declaring environment variables, see the *OS/390 DCE Administration Guide*.)

For example, if the HFS pathname of the Identity Mapping Output file is **/opt/dfslocal/home/dfskern/idmap.o**, this variable is set by the following entry in the **envar** file:

```
_IOE_MVS_IDMAP=/opt/dfslocal/home/dfskern/idmap.o
```

You can choose to have one Identity Mapping File for each server, or a single Identity Mapping File for all servers that require identity mapping (for example, multiple OS/390 DCE application support servers should share the same Identity Mapping input file).

In both cases, the declaration of the **_IOE_MVS_IDMAP** environment variable can be made in the DFS server's **envar** file.

Ensuring the _IOE_MVS_IDMAP_SAF Environment Variable Is Set to NO: The **_IOE_MVS_IDMAP_SAF** environment variable must be set to **NO** or not specified in the **dfskern** process to enable Identity Mapping.

Note: The environment variables for the **dfskern** process are only read on start-up of the **dfskern** process (not as a result of the **modify dfs,send dfskern,reload,idmap** operator command).

Modifying and Deleting Identity Mapping Entries: Edit the Identity Mapping input file to modify or delete mapping entries. Then run the **mapid** program on the Identity Mapping Input file to generate the new Identity Mapping file. For these changes to take effect, you will have to either restart the DFS server or reload the Identity Mapping Output file by using the **modify dfs,send dfskern,reload,idmap** command. See Chapter 17, “OS/390 System Commands” on page 327 for more information on the **modify** command.

Allowing Foreign Users Access to the DFS Server: If there are DFS server users from other cells (that is, ***foreign cells***), the fully-qualified DFS name of each foreign principal must be mapped to an OS/390 user ID. To define this mapping, the foreign principals are entered in the Identity Mapping input File that is used by the **mapid** utility to generate the binary Identity Mapping Output File. The **mapid** utility must translate the foreign principal name to its corresponding UUID. Hence, it must read the mapping from the foreign cell. To be able to perform this, the administrator running **mapid** must have authority to read the **/.:/sec/principal** directory in the foreign cell.

This is accomplished by having the DFS administrator add a **foreign_user** entry for the administrator (who runs **mapid**) in the ACL of the **/.:/sec/principal** directory of the foreign cell. This entry must give this administrator, read permission to this directory. For example, if the DFS server belongs to the cell **/.../cell1.tid.ibm.ca** (**/.../c=ca/o=ibm/ou=tid/cn=cell1**), and the principal name of the administrator is **cell_admin**, the DCE control program administrator interface (**dcecp** session) performed on the foreign cell is as follows:

```
$ dcecp
dcecp> acl modify /.:/sec/principal change {foreign_user /.../c=ca/o=ibm/ou=tid/cn=cell1/cell_admin r}
dcecp> exit
$
```

Access to the Identity Mapping File: The Identity Mapping file serves as the link between the DCE Security service and the local security subsystem. Access to the Identity Mapping input file and the resulting Identity Mapping Output binary file must be restricted to authorized administrators only.

The DFS server must also be given read access to the Identity Mapping file that it uses. The administrator must be given read and write access to the Identity Mapping files.

Using the RACF Identity Mapping Function Method: If you do not use the Mapping Identity File method then you can use this method.

Mapping the DCE user ID to the OS/390 user ID is a three-fold procedure:

1. Creating the RACF DCEUUIDS profile
2. Creating the RACF DCEUUIDS APPLDATA OS/390 ID entries
3. Ensuring that the **_IOE_MVS_IDMAP_SAF** environment variable is set to YES in the **dfskern** process.

Each of these procedures is described in the following sections.

Creating the RACF DCEUUIDS Profile: The RACF DCEUUIDS profile is a profile in the RACF database that allows you to store mappings of DCE user IDs to OS/390 user IDs. It is created with the following command:

```
RDEFINE FACILITY IRR.RDCERUID UACC(NONE)
PERMIT IRR.RDCERUID CLASS(FACILITY) ID(DFS1) ACCESS(READ)
SETROPTS CLASSACT(DCEUUIDS)
```

Creating the RACF DCEUUIDS APPLDATA OS/390 ID entries: RACF DCEUUIDS APPLDATA MVS ID entries specify the OS/390 user ID that the DCE cell and principal should be mapped to. RACF DCEUUIDS APPLDATA OS/390 ID entries are created with the following command:

```
RDEFINE DCEUUIDS cell_uuid.principal_uuid APPLDATA('os/390_id')
```

Ensuring _IOE_MVS_IDMAP_SAF Environment Variable is Set to YES: The **_IOE_MVS_IDMAP_SAF** environment variable must be set to YES to ensure that the RACF identity mapping facility is used. It must be set in the **dfskern** process.

Note: The environment variables for the **dfskern** process are only read on start-up of the **dfskern** process (not as a result of the **modify dfs,send dfskern,reload,idmap** operator command).

Modifying and Deleting RACF DCEUUIDS APPLDATA OS/390 ID Entries: RACF DCEUUIDS APPLDATA OS/390 ID entries may be modified by using the following command:

```
RALTER DCEUUIDS cell_uuid.principal_uuid APPLDATA('mvs_id')
```

RACF DCEUUIDS APPLDATA OS/390 ID entries may be deleted by using the following command:

```
RDELETE DCEUUIDS cell_uuid.principal_uuid
```

For these changes to take effect, you have to restart the DFS server, or reload the identity map (causing any cached mappings to be deleted by the **modify dfs,send dfskern,reload,idmap** command, Chapter 17, "OS/390 System Commands" on page 327 for more information on the **modify** commands).

Displaying RACF DCEUUDS APPLDATA OS/390 ID Entries: RACF DCEUUIDS APPLDATA OS/390 ID entries may be displayed by using the following command:

```
RLIST DCEUUIDS cell_uuid.principal_uuid
```

Removing Aggregates and Partitions from the Namespace

If you exported a DCE Local File System aggregate or non-Local File System partition, either at startup or by issuing the **dfsexport** command directly, you can issue the **dfsexport** command with the **-detach** option (not included in previous descriptions of the command's syntax) to remove the aggregate or partition from the DCE namespace. For example:

```
# dfs export -aggregate lfs1 -detach
```

In OS/390 DFS, you can issue the OS/390 **MODIFY** command to unexport all aggregates and non-Local File System partitions. For example:

```
modify dfs,start unexport
```

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.

Creating Read/Write DCE Local File System Filesets

Read/write DCE Local File System filesets are created with the **fts create** command. The **fts create** command is used to create only DCE Local File System filesets; non-Local File System filesets are created by exporting and mounting non-Local File System partitions (as described in the previous section).

Note: It is especially important on OS/390 to keep track of fileset names because certain **fts** commands require the fileset parameter **-fileset** because the pathname **-path** parameter is not supported when issuing a command on OS/390.

Before creating a read/write DCE Local File System 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 "fts aggrinfo" on page 616 for a detailed description of the **fts aggrinfo** command.)

You specify a fileset's name when you create it with the **fts create** command. A fileset's name should describe the fileset's contents. For example, the name **user.terry** describes a fileset that contains data for the user **terry**. A fileset name must also be unique within the local cell. It can be no greater than 102 characters in length, and it must contain at least one alphabetic character or an _ (underscore). (See "Fileset Names" on page 152 for more information on fileset naming rules.)

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 5, "Using ACLs and Groups" on page 75 for information about the interaction between ACLs and UNIX mode bits and for suggestions for initial ACLs); and
- Assigns a default quota of 5000 kilobytes to the fileset.

The following sections describe the steps involved in creating a read/write DCE Local File System fileset. 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 Local File System filesets.

Creating a Read/Write Fileset

After creating a read/write DCE Local File System 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 its 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 Local File System filesets.

Creating and Mounting a Read/Write Fileset: To create and mount a read/write fileset, perform the following steps:

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. You must also 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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) ACL permissions for the directory where 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 11, "Managing Filesets" on page 197 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.

   Note: If the parent directory resides in an OS/390 fileset, it must be contained in a DCE Local File System Aggregate or an HFS Aggregate. (That is, you cannot create a DFS mount point in an RFS fileset.)

Resetting the Fileset Quota: To reset the quota for the new read/write fileset, perform the following steps:

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 **-fileset** *name* or *ID* option is the name or ID 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 fileset can occupy, including files referenced by this fileset but housed in another fileset type of this fileset. Specify the value in kilobytes (a value of 1024 kilobytes equals 1 megabyte).

Creating Read-Only DCE Local File System Filesets

Replication is the process of creating read-only copies (replicas) of a read/write DCE Local File System 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 housing the fileset becomes unavailable, the fileset can usually still be accessed from another machine. Replication is not available for non-Local File System 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 popular 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 housing 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 Local File System filesets:

- *Release Replication* 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 housing a read-only replica then replaces the copy of the replica on its machine with the new replica stored on the same machine as the read/write fileset. The read-only replicas do not change until you explicitly issue the **fts release** command to update them.

  Use Release Replication if the fileset seldom changes or if you need to closely track the distribution of new replicas.
- *Scheduled Replication* requires you to specify replication parameters that control the length of time between the automatic release of new replicas. The Replication Server on each machine housing a read-only site updates the read-only replica on its machine according to time intervals that you supply.

  Use Scheduled Replication if you prefer to have the system release updates of replicas at regular intervals and you do not need to closely track the distribution of new replicas.

A read/write fileset can be replicated with only one of the two types of replication at any one time. Regardless of the type of replication in use, an immediate update of the read-only copies of a source fileset can be requested at any time with the **fts update** command. The command can be used to update all replicas or only the replica at a specific site. It has no effect on the replication type and parameters that are defined for the read/write fileset.

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. After it is set, replication information is stored in the FLDB entry for 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. The following section provides more information on these parameters.

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. The site definitions are recorded in the entry for the fileset in the FLDB. It is not possible to place multiple read-only copies of the same fileset on a single File Server machine.

Although read-only replicas do not reside at the replication sites until the source fileset is first replicated, the status flag of the read-only version in the FLDB is changed from **invalid** to **valid** once a replication site is defined. It is assumed that replicas will be placed at the sites shortly after the sites are defined.

All read-only copies of a read/write fileset share the same fileset ID number. The number, which is reserved in the FLDB 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.

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 “Creating Backup DCE Local File System Filesets” on page 190 for more information about the structure of backup filesets and the sharing of data among the different types of DCE Local File System filesets.

The FLDB can record a maximum of 16 sites for all versions of a fileset combined, a site being a specific File Server machine and aggregate. The read/write version and backup version (if it exists) of a fileset share a single site definition. If you define a replication site for a fileset at the same site as its read/write and backup versions (which is recommended for any fileset, especially one that uses Release Replication), you can then define 15 more 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.

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 Local File System fileset;
- **root.dfs** was mounted with an explicit read/write mount point when the cell was configured; and
- **root.dfs** is replicated.

See “Setting Up the Root Fileset” on page 59 for information about configuring **root.dfs** to support replication.

Preparing for Replication

The following prerequisites must be met before you can replicate a read/write fileset:

- Each File Server machine that is to house a replica of the fileset must have a server entry in the FLDB. (See “Creating a Server Entry for a File Server Machine” on page 161 for information on creating server entries.)
- A Replication Server (**repserver** process) and a Fileset Server (**ftserver** process) must be running on each File Server machine that is to house a replica of the fileset. Use the **bos status** command to verify that the **repserver** and **ftserver** processes are running on the machine at each replication site. (You can also use the **fts statrepserver** command, which is described in “Displaying Replication Status” on page 189 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 replication sites on that machine. Use the **fts lsft** command to check the size of the read/write fileset. The **fts aggrinfo** and **fts lsft** commands are described in detail in the following section. 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 sections describe how to set the replication type and parameters and how to add replication sites.

Replication Type and Parameters: Before defining any replication sites, you must use the **fts setrepinfo** command to indicate the type of replication to be used and to set the replication parameters for the fileset to be replicated. When initially defining the type of replication to be used with a fileset, include either the **-release** option or the **-scheduled** option with the command to indicate the type of replication to be used.

Most of the options available with the **fts setrepinfo** command determine the replication parameters to be associated with the fileset (the primary exceptions are the **-change** and **-clear** options, 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 housing the replicas. Descriptions of the replication parameters follow; each parameter is set with an option of the same name. (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 maximum 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 how current a read-only replica is, which it communicates to the Cache Manager through the File Exporter. For scheduled replication, the verification is made with respect to the read/write source; for release replication, the verification is made with respect to the read-only fileset itself.

FailAge Specifies the amount of time the Cache Manager distributes data cached from a read-only replica if that data cannot be verified as current. The difference between FailAge and MaxAge is the amount of time the Cache Manager continues to distribute data cached from a read-only replica after that data cannot be verified as current. 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 referenced 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 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. The Cache Manager sends keep-alive messages more frequently than the ReclaimWait interval to prevent the File Exporter from reclaiming storage space that was occupied by deleted files.

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 currency 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 9 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) listed in the table when setting replication parameters.

Table 9. Descriptions of Replication Parameters

Parameter	**Default**	**Release Replication**	**Scheduled Replication**
MaxAge (**fts setrepinfo**)	2 hours	Required only if FailAge is specified	Required only if FailAge, MinRepDelay, or DefaultSiteAge is specified
FailAge (**fts setrepinfo**)	1 day or twice MaxAge, whichever is larger	Optional	Required only if MinRepDelay or DefaultSiteAge is specified
ReclaimWait (**fts setrepinfo**)	18 hours	Required only if FailAge is specified	Required only if FailAge, MinRepDelay, or DefaultSiteAge is specified
MinRepDelay (**fts setrepinfo**)	5 minutes or one-quarter of DefaultSiteAge, whichever is smaller	Not applicable	Required only if FailAge or DefaultSiteAge is specified
MaxSiteAge (**fts addsite**)	DefaultSiteAge	Not applicable	Required only if DefaultSiteAge is not specified
DefaultSiteAge (**fts setrepinfo**)	One-quarter of MaxAge	Not applicable	Optional

The system uses the previous guidelines to calculate default values for each of the parameters unless you specify

- FailAge for Release Replication; or
- FailAge, MinRepDelay, or DefaultSiteAge for Scheduled Replication.

Once you specify one of these parameters, the system no longer performs any default calculations; you must specify values for all applicable parameters. The exception is DefaultSiteAge for Scheduled Replication, which is always optional. However, if the other parameters specified with the **fts setrepinfo** command are supplied, the system does not calculate a default value for DefaultSiteAge; you must specify the MaxSiteAge for each replication site with the **fts addsite** command. Also, because the MinRepDelay, MaxSiteAge, and DefaultSiteAge parameters do not apply to Release Replication, they are recorded but otherwise ignored if they are specified for a fileset that uses Release Replication. (They are used if the fileset's style of replication is ever changed to Scheduled Replication.)

Setting Replication Type and Parameters: To set a fileset's replication type and parameters, perform the following steps:

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 9 on page 185 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).

   ```
   $ 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 Local File System 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 either this option or the **-scheduled** option.
   - The **-scheduled** option specifies that Scheduled Replication is to be used with the fileset. When initially setting the replication type, use either this option or 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.
   - 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 with the **fts addsite** command if the **-maxsiteage** option is omitted from that command.
   - The **-clear** option removes all replication parameters previously defined for the fileset.

Listing Replication Type and Parameters: You can use the **fts lsfldb** command and the **fts lsft** command to list information on the replication type and replication parameters defined for a fileset. See Chapter 11, "Managing Filesets" 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 {file name | directory_name} | -fileset {name | ID}}] [-server machine]
```

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 the 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 being 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 the previous section for details about the syntax of the **fts setrepinfo** command. See Table 9 on page 185 for information about which of the command's options apply to the two types of replication and the dependencies between the parameters.

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.

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, with Release Replication, you must choose the File Server machine on which the read/write fileset resides as the first replication site. Finally, 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.

Adding a Replication Site: To add a replication site for a fileset, perform the following steps:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset

resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Select a replication site for the read/write fileset that is to be replicated. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate (the **fts lsft** command can be used to check the size of the read/write fileset).

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

3. Enter the **fts addsite** command to add the replication site. If Release Replication is being used, you must define the first replication site on the same File Server machine as the read/write fileset.

   ```
   $ fts addsite -fileset {name | ID} -server machine -aggregate name [-maxsiteage interval]
   ```

   The **-maxsiteage** *interval* option can be used to override the DefaultSiteAge set with the **fts setrepinfo** command.

   Repeat the **fts addsite** command for each replication site you want to define for the fileset.

Removing a Replication Site: To remove a replication site for a fileset, perform the following steps:

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. (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.) 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. (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 a fileset.

Creating Read-Only Filesets

The following sections describe how to

- Use Release Replication to replicate a fileset with the **fts release** command;
- Use Scheduled Replication to replicate a fileset; and
- Request that the Replication Server perform an immediate update of all replicas of a fileset or of only the replica at a given site with the **fts update** command.

The operations described in this section assume that the preparatory steps described in the previous section have all been performed. The following section provides information about how to check the status of the Replication Server on a specific machine or the status of replicas of a fileset at one or all replication sites.

Using Release Replication to Create Read-Only Filesets: For a fileset that is using release replication, perform the following steps 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. You must also 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 the 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 place a copy of this read-only fileset at their respective sites.

   ```
   $ 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.

Using Scheduled Replication to Create Read-Only Filesets: Scheduled Replication takes place automatically, based on the parameters previously established with the **fts setrepinfo** command. You do *not* have to explicitly enter a command to initiate Scheduled Replication.

Requesting an Immediate Update of Read-Only Filesets: Issue the **fts update** command to request an immediate update of replicas at a specific site or all replication sites:

```
$ fts update -fileset {name | ID} {-all | -server machine}
```

- The **-fileset** *name* or *ID* option is the name or ID of the 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 either the **-all** or **-server** option.
- The **-server** *machine* option specifies the DCE pathname of a specific File Server machine where the replica of **-fileset** to be updated is stored. Specify the File Server machine using the machine's DCE pathname, the machine's hostname, or the machine's IP address. Use either this option or the **-all** option.

Displaying Replication Status

The operations described in the following sections allow you to do the following tasks.

- 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 status 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. If replicas of a fileset are not current, use the **fts update** command to update them.

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 **-server** is to be displayed. The additional information includes the status of each replica that is managed by the Replication Server.

Displaying the Status of Read-Only Filesets: Use the **fts lsreplicas** command to verify the status 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 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 either the **-all** or **-server** option.
- The **-server** *machine* option specifies the DCE pathname of a specific File Server machine where replicas of **-fileset** are to be checked. Specify the File Server machine using the machine's DCE pathname, the machine's hostname, or the machine's IP address. Use either this option or the **-all** option.

Creating Backup DCE Local File System Filesets

A backup fileset is a single copy of the read/write version of a DCE Local File System fileset; backup versions of non-Local File System filesets cannot be created. A backup fileset is stored at the same site as the read/write version and can serve as an on-line backup copy of a fileset. This process has the following purposes:

- It is the first step in using the Backup System to copy a DCE Local File System 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 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 their state at the time of the backup without your help.

  Note: If you do create and mount backup filesets for your users, you need to explain to them how often you will make the backups and remind them that the data in their backup filesets cannot be changed. They can, however, copy the data to a directory in a read/write fileset and use it there.

An Overview of Backup Filesets

Although backup and read-only filesets are created in a similar manner, they each 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 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 that point 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 result is significant disk space savings. (See “Data Sharing Among the Different Types of DCE Local File System Filesets” on page 151 for more information on data sharing among the different types of DCE Local File System 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. 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, which is the version that existed in the read/write fileset when the backup was made.

The backup fileset normally resides at the same site (File Server machine and aggregate) as the read/write source. It has the same name as the read/write version, with the addition of a **.backup** extension.

Backup Options

You can use the **fts clone** command to back up a single read/write DCE Local File System fileset; you can use the **fts clonesys** command to back up multiple read/write DCE Local File System 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; and
- 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, 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.

The **bos create** command on OS/390 DFS bosserver is slightly modified to allow output redirection to a DD allocated to the DFS procedure. For example:

```
$ bos create /.../abc.com/hosts/fs3 backupusers cron \
"IOEFTS clonesys -prefix user -localauth dd:clonesys" 5:30
```

Creating and Mounting Backup Filesets

To create and mount backup filesets, perform the following steps:

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. You must also 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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) 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.

   See also, the **modify dfs,send dfskern,clonesys** OS/390 system command on page 329.

4. Enter the **fts crmount** command to create a mount point for each backup fileset. The contents of filesets are inaccessible until you perform this operation.

   ```
   $ fts crmount -dir directory_name -fileset name.backup
   ```

   - The **-dir** *directory_name* option is the location for the root directory of the backup fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.
   - The **-fileset** *name***.backup** option is the full name of the backup fileset, with the **.backup** extension.

     Repeat the **fts crmount** command for each backup fileset created in the previous step. Note that you do not need to create a mount point for a backup fileset; for example, the DFS Backup System does not require a mount point to locate a backup fileset.

Using Mount Points

To make a DCE Local File System or non-Local File System 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 at a directory location; they are not accessible if the fileset is not mounted. After the mount point is created, the fileset is automatically mounted; you never need to explicitly mount it again.

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. During file retrieval, the Cache Manager traverses the file's complete pathname, continuing until it finds the file. When the Cache Manager encounters a mount point, it reads the mount point to learn which fileset is associated with that directory location and contacts the Fileset Location Server to locate the fileset in the cell. It then finds, as the root of the fileset, an actual directory structure and gives the directory the same name as the mount point. The fileset's root directory provides a pathname to all of the files, subdirectories, and mount points in the fileset. The Cache Manager checks the next element in the pathname in the root directory and interprets any further mount points it finds until it reaches the fileset that contains the requested file.

It is recommended that you *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 (only in a very limited number of troubleshooting and testing situations). Remove the extraneous mount points as soon as they are no longer necessary.

A mount point can be characterized by its fileset type (read/write, read-only, or backup) or by its mount point type (regular or read/write).

The following section describes the different types of mount points in greater detail. 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 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 # (pound 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 referenced by a removed mount point remains, but its contents are not accessible until another mount point is created.

Types of Mount Points

There are several different categories of mount points. A mount point can be distinguished by its fileset type and its mount point type. The following sections describe the characteristics of mount points.

Fileset Type: The first characteristic of a mount point determines which type of fileset is named in the mount point. A read-only or backup fileset has a **.readonly** or **.backup** extension; a read/write fileset has no extension.

When a mount point names a fileset with a **.readonly** or **.backup** extension, the Cache Manager uses only the specified 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 that has no extension).

Mount Point Type: The second characteristic determines the mount point type. 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 uses 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. 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 attempts to access a read-only version of the fileset first. If no read-only versions exist, the Cache Manager attempts to access the read/write version of the fileset. If one or more read-only versions exist but all are unavailable, the Cache Manager cannot access the fileset (it does not attempt to access the read/write version).

Regular mount points allow the Cache Manager to retrieve files better than other types of mount points because they allow the Cache Manager to access read-only filesets whenever possible. There are normally several different instances of a read-only fileset, but only one copy 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 fileset when it encounters a regular mount point if no other versions exist or if it is already on a read/write path traversal.

The least common type of mount point is the global root mount point. This type of mount point is used to mount the root of the DCE global namespace. (Because it is maintained for backward compatibility with other file systems, it is not normally used. This documentation provides no examples of its use.)

Manipulating Mount Points

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

Creating a Mount Point: To create a mount point, perform the following steps:

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 Local File System fileset, you must have WRITE (**w**), EXECUTE (**x**), CONTROL (**c**), and INSERT (**i**) 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-Local File System fileset, you must have **w** and **x** permissions for the directory.

   Note: You cannot create a mount point in an RFS fileset.

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 it 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 at **/.../abc.com/fs/usr**.

```
$ cd /.../abc.com/fs/usr
$ ls
```

```
pat      terry
```

The **fts crmount** command is then used to mount the fileset named **user.vijay** at **/.../abc.com/fs/usr**. Because the **-rw** option is omitted, the fileset is mounted with a regular mount point.

```
$ fts crmount /.../abc.com/fs/usr/vijay user.vijay
$ ls
```

```
pat      terry    vijay
```

Listing a Mount Point: To list a mount point, perform the following steps:

1. Verify that you have the READ (**r**) 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 Local File System 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:

   ```
   $ fts lsmount vijay
   ```

   ```
   'vijay' is a mount point for fileset '#user.vijay'
   ```

Removing a Mount Point: To remove a mount point, perform the following steps:

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 Local File System fileset, you must have WRITE (**w**), EXECUTE (**x**), and DELETE (**d**) 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-Local File System fileset, you must have **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 in the DCE namespace.

   ```
   $ ls
   ```

```
pat      terry    vijay
$ fts delmount vijay
$ ls
pat      terry
```

Chapter 11. Managing Filesets

Important Note to Users

The following list notes exceptions to the contents of this chapter for DFS users on the OS/390 UNIX operating system:

- This guide uses the term *non-Local File System* to refer to non-DCE Local File System filesets and aggregates. For the OS/390 DFS implementation, non-Local File System and non-DCE Local File System refer to OS/390 Hierarchical File System (HFS) data sets.
- The terms *partition* and *disk partition* as used in this chapter may refer to OS/390 Hierarchical File Systems (HFS) when referring to non-Local File System or a ***Logical Volume*** when referring to DCE Local File System.
- Any statements that specify the user must login as **root**, refers to logging in with a user ID which has a **UID = 0**.

This section 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 10, “Making Filesets and Aggregates Available” provides introductory information about filesets and 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 Local File System and non-Local File System filesets.

An Overview of Fileset Terminology

DCE Local File System aggregates are similar to HFS file systems that are found in the OS/390 UNIX operating systems. However, a DCE Local File System aggregate also supports specialized fileset-level operations, such as quota-checking and cloning, and low-level operations, such as logging of metadata. Each DCE Local File System aggregate exported to the DCE namespace can house multiple filesets; filesets stored on DCE Local File System aggregates are referred to as DCE Local File System filesets. Each exported non-Local File System disk partition can house only a single fileset; file systems stored on non-Local File System partitions are referred to as non-Local File System filesets. Most of the specialized features supported for DCE Local File System filesets are not supported for non-Local File System filesets.

In DFS, only a single type of each non-Local File System fileset is available: the version that was made available when the partition on which it resides was exported. However, three types of each DCE Local File System fileset are available:

- A read/write version of a DCE Local File System fileset contains modifiable versions of the files and directories in that fileset. Every DCE Local File System 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 Local File System 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 Local File System 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 Local File System and non-Local File System 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 Local File System 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 Local File System 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 10, "Making Filesets and Aggregates Available" for information about DCE Local File System filesets, non-Local File System filesets, and how the two types of filesets are created.

Standard Options and Arguments

When using the **fts** commands to create, manipulate, or delete filesets, you can often add the **-verbose** option to receive detailed information from the **fts** program about its actions as it executes the command. This can be particularly helpful if an operation fails for a reason that you do not understand. The amount of additional information produced by the **-verbose** option varies for different commands.

The following options and arguments are common to many of the commands described in this section. If an option or argument is not described with a command in the text, a description of it appears here. See Chapter 19, "Distributed File Service Commands" on page 401 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 hostname (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/lfs1**), 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 **/opt/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 6, “Using Administrative Lists and Keytab Files” 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 10, “Making Filesets and Aggregates Available” for information about DFS server principals.)

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

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

Listing Fileset Information

The following commands are available for listing information about filesets:

- The **fts lsfldb** command is used to list information about filesets from the FLDB.
- The **fts lsheader** command is used to list information from fileset headers on File Server machines and aggregates.
- The **fts lsft** command is used to list information from both the FLDB entry and the fileset header of a specific fileset.

If you know only the name of a file or directory that is housed in a fileset, you can use the **fts lsquota**, **fts lsft**, and **cm whereis** commands to access information about the fileset. (See “Determining Other Fileset Information” on page 203 for complete details on using these commands with file or directory names.)

Listing FLDB Information

The **fts lsfldb** command is used to display information from entries in the FLDB for both DCE Local File System and non-Local File System 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.

Listing the FLDB

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 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 Local File System fileset:

```
$ 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>
```

Listing Fileset Header Information

The **fts lsheader** command displays the fileset header from every DCE Local File System 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 Local File System fileset, use the **fts lsft** command.

Non-Local File System filesets do not have DCE Local File System 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 **/opt/dcelocal/var/dfs/dfstab** file on a File Server machine.

Listing Fileset Headers

To list information from fileset headers, perform the following steps:

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
- Its fileset ID number
- Its type (`RW` for read/write, `RO` for readonly, or `BK` for backup)
- Its allocation usage and quota usage, in kilobytes
- Its status (`On-line`, `Off-line`, or an error indicator)
- The total number of filesets that are online, the total number of filesets that are offline, and the total number of filesets that are busy

Following is an example of the output from the command when it is run with no options:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1

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 Local File System (`Local File System`) or non-Local File System fileset
- Information about the state of the fileset

- The ID numbers of the parent, clone, and backup filesets that are related to the fileset
- The ID numbers of the low-level backing and low-level forward filesets that are related to the fileset
- The version number of the fileset
- The allocation and allocation usage, in kilobytes, of the fileset
- The quota and quota usage, in kilobytes, of the fileset
- The day, date, and time when the fileset was created (replicated or cloned for a read-only or backup fileset)
- The day, date, and time when the contents of the fileset were last updated (this is the same as the creation time for a read-only or backup fileset)

Following is an example of the output from the command when it is executed with the **-long** option:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1 -long

Total filesets on server fs3 aggregate lfs1 (ID 10): 16
user.terry 0,,196953 RW LFS     states 0x10010005   On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    429496729 K alloc limit;      1252 K alloc usage
        15000 K quota limit;      9340 K quota usage
    Creation Tue Oct 15 16:45:16 1991
    Last Update Fri Nov 22 11:36:00 1991

user.wvh 0,,196956 RW LFS     states 0x10010005  On-line
  ⋮
Total filesets on-line 15; total off-line 1; total busy 0
```

Listing FLDB and Fileset Header Information

You can use the **fts lsft** command to display information from both the FLDB and the fileset header for a single fileset. The output from the **fts lsft** command consists of the output from the **fts lsheader** command with the **-long** option followed by the output from the **fts lsfldb** command. You can indicate the fileset about which information is to be displayed in one of two ways: by specifying the name of a file or directory that is stored in the fileset with the **-path** option, or by specifying the name or fileset ID number of the fileset with the **-fileset** option.

Because the **fts lsft** command retrieves information from the fileset header, you can examine the read-only or backup version of a DCE Local File System 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 Local File System 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 Local File System 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 Local File System 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 “Removing DCE Local File System Filesets” on page 220 for more information about deleting DCE Local File System—and non-Local File System— filesets.)

Because non-Local File System filesets do not have fileset headers, the **fts lsft** command displays much less fileset header information for non-Local File System filesets than it does for DCE Local File System 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 Local File System 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
number of sites: 2
  Sched repl: maxAge=2:00:00;
        failAge=1d0:00 :00;
  reclaimWait=18:00:00; minRepDelay=0:05:00;
defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Determining Other Fileset Information

The following sections 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. These sections include descriptions of the following commands:

- The **fts lsquota** command, is used to determine the name of a fileset from the name of a file or directory that it houses.

Note: The primary usage of the **fts lsquota** command is to list fileset quota information, not to determine fileset names. (See "Setting and Listing Fileset Quota" on page 209).

- The **fts lsft** command, 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** and **cm whereis** commands, are used to learn the location of a fileset from its name or ID number or from the name of a file or directory that it houses.

The **cm whereis** command is from the DFS **cm** command suite. (See Chapter 13, "Configuring the Cache Manager" on page 251 for more information about **cm** commands used to configure the Cache Manager, see "cm" on page 543 for more detailed information about all **cm** commands.)

Learning Fileset Names

To determine fileset names from file or directory names, 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 unknown fileset. You can include multiple files or directories from different filesets. Omit this option to learn the name of the fileset that houses the working directory.

The **fts lsquota** command produces a single line of output for each fileset that houses a named file or directory. Each line begins with the name of the fileset to which the information corresponds. (See "Setting and Listing Fileset Quota" on page 209 for a complete description of the command's output.)

Following is an example of the output from this command:

```
$ fts lsquota /.../abc.com/fs/usr/terry

Fileset Name     Quota   Used   % Used   Aggregate
user.terry       15000   5071      34%     86% = 84538/98300 (LFS)
```

Learning Fileset ID Numbers

To learn a fileset ID number from a file or directory name, 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 an unknown fileset. Omit this option to learn the ID number of the fileset that houses the working directory.

The **fts lsft** command begins its output with the name and ID number of the fileset that houses the named file or directory. (See "Listing FLDB and Fileset Header Information" on page 202 for a description of the command's output.)

The following example displays the fileset ID number 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>
```

To learn a fileset ID number from a fileset name, enter the **fts lsft** command with the **-fileset** option:

```
$ fts lsft -fileset {name | ID}
```

The following example displays the fileset ID number of the fileset whose name is **user.terry**:

```
$ fts lsft -fileset user.terry

user.terry 0,,196953 RW LFS     states 0x10010005 On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
  ⋮
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Learning Fileset Locations

To learn the location of a fileset from the fileset name or ID number, enter the **fts lsfldb** command with the **-fileset** option:

```
$ fts lsfldb -fileset {name | ID}
```

The **fts lsfldb** command concludes its output with a list of the File Server machines that house existing versions of the fileset. (See "Listing FLDB Information" on page 199 for a thorough description of the command's output.)

Following is an example of this command and its output:

```
$ 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>
```

To learn the locations of filesets from file or directory names, 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 an unknown fileset. You can include multiple files or directories from different filesets. Omit this option to learn the location of 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.

Following is an example of the output from this command:

```
$ cm whereis /.../abc.com/fs/usr/terry

The file '/.../abc.com/fs/usr/terry' resides in the cell
'abc.com', in fileset 'user.terry', on host fs3.abc.com.
```

Listing Aggregate and Partition Information

The following commands are available for listing information about any aggregate or partition (non-Local File System aggregate) that is exported to the DCE namespace:

- The **fts lsaggr** command is used to list all exported aggregates or partitions on a File Server machine.
- The **fts aggrinfo** command is used to list information about the amount of disk space available on a specific aggregate or partition or on all exported aggregates and partitions on a File Server machine.

These commands are especially useful when exporting additional aggregates or partitions from a machine, when creating read/write or read-only filesets on an aggregate, or when moving filesets between machines.

Listing Aggregates and Partitions

The **fts lsaggr** command is used to list the following information about each exported aggregate or partition on a File Server machine. The information is specified for each aggregate and partition in the **/opt/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-Local File System partitions and two DCE Local File System aggregates are exported from the File Server machine named **fs1**:

```
$ fts lsaggr /.../abc.com/hosts/fs1

There are 2 aggregates on the server /.../abc.com/hosts/fs1 (fs1.abc.com)
                lfs1 (/dev/lfs001): id=1     (LFS)
                hfs1 (/dev/ufs001): id=101   (Non-LFS)
```

Listing Disk Space on Aggregates and Partitions

The **fts aggrinfo** command lists information about the total amount of disk space and the amount of disk space that is currently available on exported aggregates and partitions. It can be used to obtain size information about a specific aggregate or partition or about all of the exported aggregates and partitions on a File Server machine. The command displays the following information about each aggregate and partition:

- The file system type (DCE Local File System or non-Local File System).
- The aggregate name.
- The device name.
- The number of kilobytes of disk space that is currently available on the aggregate or partition.
- The total number of kilobytes on the aggregate or partition.
- The number of kilobytes, if any, of reserved disk space on the aggregate or partition. DCE Local File System 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-Local File System 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-Local File System partitions, exported DCE Local File System aggregates, and locally mounted DCE Local File System 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 **dcefvt8**:

```
$ fts aggrinfo dcefvt8

LFS aggregate lfs6 (/dev/lfs006): 1095 K free out of total 1296 (136 reserved)
LFS aggregate lfs8 (/dev/lfs008): 12739 K free out of total 12960 (1432 reserved)
LFS aggregate lfs10 (/dev/lfs010): 18606 K free out of total 18872 (2000 reserved)
LFS aggregate lfs20 (/dev/lfs020): 6335 K free out of total 6480 (712 reserved)
LFS aggregate lfs21 (/dev/lfs021): 6335 K free out of total 6480 (712 reserved)
Non-LFS aggregate ufs200 (/dev/ufs200): 3312 K free out of total 3600
Non-LFS aggregate ufs210 (/dev/ufs210): 10052 K free out of total 60480
```

Increasing the Size of a DCE Local File System Aggregate

Important Note to Users

This section includes information on using the **growaggr** command to increase the size of a Logical Volume. In OS/390 DFS, **growaggr** can be run as a batch job, as a TSO/E command, or from the OS/390 shell. The *xxx*.SIOESAMP(GROWAGGR) member (where *xxx* is installation dependent) has sample JCL for running this program in batch.

For more information on running **growaggr** from TSO/E, or from the OS/390 shell, refer to “growaggr” on page 731.

DCE Local File System aggregates are created with the **newaggr** batch program. The initial size of an aggregate is specified with the **-aggrsize** option. See Chapter 10, “Making Filesets and Aggregates Available” on page 147 for more information about using the newaggr command or see “newaggr” on page 737.

The **growaggr** command is used to increase the size of an existing DCE Local File System aggregate. You can use **growaggr** to enlarge the aggregate within a single linear data set, or enlarge it into an additional data set. 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 Local File System blocks greater than the current size of the aggregate. (The number of bytes in a DCE Local File System 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 Local File System aggregate, perform the following steps:

1. Be sure that the DFS kernel (**dfskern**) is active on the OS/390 UNIX system.
2. The aggregate to be enlarged must *not* be exported to DFS. If it is exported, enter the following **dfsexport** command from OS/390 UNIX to detach the aggregate:

   ```
   dfsexport lfs1 -detach
   ```

3. Specify the **growaggr** command to increase the size of the aggregate. See “growaggr” on page 731 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 Local File System aggregate to be grown. These identifiers are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file. If you are adding an additional linear data set, you must include the name of the data set in the logical volume's sequential file *before* issuing the command.

   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 Local File System 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.

4. Log in as **root** on the machine on which the aggregate that is to be enlarged resides. The **growaggr** command must run as **UID = 0** to access the **devtab** file.
5. Process the **growaggr** command by submitting member GROWAGGR in DCE.*version*.JCL. This is a two step process. The first step of this process will create a new linear data set. If you are adding an additional linear data set, update the data set name and space parameters before continuing. If a new linear data set is not needed, remove the linear data set created in this step. The second step of this process will execute the **growaggr** command.
6. If you want to export the aggregate, enter the following command:

   ```
   dfsexport lfs1
   ```

Setting and Listing Fileset Quota

By default, every newly created DCE Local File System fileset has a quota of 5000 kilobytes (1024-byte units). The **fts setquota** command can be used to modify the quota of a DCE Local File System fileset. The **fts lsquota** command can be used to examine the available quota and quota usage for a DCE Local File System fileset.

Because it does not represent the amount of physical data stored on the fileset, the quota of a DCE Local File System fileset can be larger than the size of the aggregate on which the fileset resides. Similarly, the combined quota limits of all filesets on a DCE Local File System 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. Also, 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.

The size of a non-Local File System fileset is equivalent to the size of the partition on which it is stored. In the UNIX operating system, you can use the **df** command to determine the size of a non-Local File System partition. The **df** command can also be used to check the size of an exported DCE Local File System aggregate, but it cannot be used to display the size of a DCE Local File System fileset unless the fileset is mounted locally. Although you can use the **fts lsquota** command in DFS to check the space that is used and available on a non-Local File System fileset, you cannot use the **fts setquota** command to set the quota of a non-Local File System 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 Local File System and non-Local File System filesets. For a DCE Local File System 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 Local File System aggregate.

For a non-Local File System fileset, the command displays the following information:

- The name of the fileset.

- For HFS, the quota is the actual physical size of the non-Local File System aggregate when it was created. It cannot change by using the **fts setquota** command. The usage and percentage used also displays valid information.
- For RFS, the quota and the used values are dummy values (120000, 60000), and the percentage used and aggregate are both 50%. This occurs because the DFS file server cannot accurately calculate OS/390 dataset space utilization.
- 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-Local File System aggregate (partition).

Note again that the UNIX **df** command can be used to display the same information for the partition on which a non-Local File System fileset resides.

Setting Quota for a DCE Local File System Fileset

To set the quota for a DCE Local File System fileset, perform the following steps:

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 on 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 fileset can occupy, including files referenced 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.

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 on each fileset whose quota you want to display. You can include multiple files or directories from different filesets. 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 quota information about the fileset that contains the working directory.

Following is an example of this command and its output.

```
$ fts lsq -file charlieft1 charlieft2 hfsseta hfssetb rfsset

Fileset Name          Quota    Used  % Used   Aggregate
charlieft1             5000       9     0%     1% = 221/12960 (LFS)
charlieft2             5000       9     0%     1% = 221/12960 (LFS)
hfsseta                3600     288     8%     8% = 288/3600 (non-LFS)
hfssetb               60480   50428    83%    83% = 50428/60480 (non-LFS)
rfsset               120000   60000    50%    50% = 60000/120000 (non-LFS)
```

Setting Advisory RPC Authentication Bounds for Filesets

You can set upper and lower advisory RPC authentication bounds for any DCE LFS fileset. These bounds 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.

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 fileset 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_c_protect_level_default** or **default**: Use the DCE default authentication level.
- **1** or **rpc_c_protect_level_none** or **none**: Perform no authentication.

- **2** or **rpc_c_protect_level_connect** or **connect**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **3** or **rpc_c_protect_level_call** or **call**: Authenticate only at the beginning of each RPC received.
- **4** or **rpc_c_protect_level_pkt** or **pkt**: Ensure that all data received is from the expected host.
- **5** or **rpc_c_protect_level_pkt_integrity** or **pkt_integrity**: Authenticate and verify that none of the data transferred has been modified.
- **6** or **rpc_c_protect_level_pkt_privacy** or **pkt_privacy**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.
- **7** or **rpc_c_protect_level_cdmf_priv** or **cdmf_priv**: Perform authentication as specified by all of the previous levels (except 6) and also encrypt each RPC argument value. This level provides a lower level of packet privacy than **rpc_c_protect_level_pkt_privacy**.

The following command sets the authentication values as follows:

- The maximum authentication level for communication with Cache Managers in the local cell is set to packet integrity.
- The minimum authentication level for communication with Cache Managers in the local cell is set to packet.
- The maximum authentication level for communication with Cache Managers in foreign cells is set to packet security.
- The minimum authentication level for communication with Cache Managers in foreign cells is set to packet security.

```
$ fts setprotectlevels -fileset richland.12 -maxlocalprotectlevel 5 \
                       -minlocalprotectlevel 4 -maxremoteprotectlevel 6 \
                       -minremoteprotectlevel 6
```

Renaming Filesets

You can use the **fts rename** command to change the name of the read/write version of any DCE Local File System or non-Local File System 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, perform the following steps:

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 WRITE (**w**), EXECUTE (**x**), INSERT (**i**), and DELETE (**d**) ACL permissions for the directory in which you will 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 10, “Making Filesets and Aggregates Available” for more information on fileset naming conventions.)

4. Issue the **fts delmount** command to remove the mount point that indicates the fileset's old name:

   ```
   $ fts delmount -dir directory_name...
   ```

   The **-dir** *directory_name* option is the name of each mount point you want to remove.

5. Issue the **fts crmount** command to create a new mount point that indicates the fileset's new name:

   ```
   $ fts crmount -dir directory_name -fileset {name | ID}
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the current working directory.

   The **-fileset** *name* or *ID* option is the name or ID number of the fileset to be mounted. Use the name you specified with the **fts rename** command in the earlier step.

Moving DCE Local File System Filesets

The **fts move** command allows you to move read/write versions of DCE Local File System filesets between aggregates on the same File Server machine or between File Server machines. Read/write DCE Local File System filesets are the only types of filesets you can move. You cannot move read-only or backup DCE Local File System filesets and you cannot move non-Local File System filesets.

You can use the **fts move** command to relocate the read/write filesets from a machine or disk that must be removed for repair. You can also use the command to move read/write filesets from an aggregate that is full or from a File Server machine that is overloaded with requests for frequently accessed data.

When you move a fileset, you do not need to modify the entry of the fileset in the FLDB in any way. Information in the FLDB entry of the fileset is updated automatically. Also, because mount points are location-independent, you do not need to change the mount point for a fileset when you move the fileset to a new site.

If you move a read/write fileset for which a backup version exists, the backup version is automatically deleted from the read/write site; use the **fts clone** command to create a backup version at the new site. If you move a read/write fileset that uses Scheduled Replication, the move has no effect on the fileset replicas. If you move a read/write fileset that uses Release Replication, the move has no effect on most of the fileset replicas, but the replica that resides on the same File Server machine as the read/write fileset receives special treatment. The effect of the move operation on this replica differs depending on whether the read/write fileset is moved to an entirely different File Server machine or merely to a different aggregate on the same File Server machine, as follows:

- **If you move the read/write fileset to a different File Server machine** — the command removes the existing replication site and replica from the machine from which the fileset is moved. It then creates a new replication site and replica on the File Server machine and aggregate to which the fileset is moved. Because it is created on the same File Server machine and aggregate as the read/write fileset, the new replica is created as a clone of the read/write fileset.

  The machine to which the fileset is to be moved cannot already house a replication site and replica of the fileset. If it does, you must use the **fts rmsite** command to remove the existing replication site and replica from the destination machine *before* using the **fts move** command; otherwise, the command fails.

The **fts rmsite** command removes the replication site from the FLDB entry for the fileset and instructs the Replication Server at the site to remove the read-only fileset. Deletion of the read-only fileset is usually instantaneous, but if a problem prevents the Replication Server from removing the replica, the subsequent **fts move** command fails. You can use the **fts lsft** command to confirm that the replica has been deleted before you issue the **fts move** command.

- **If you move the read/write fileset to a different aggregate on the same File Server machine** — the effect of the command depends on the aggregates involved in the operation:
  - If the replica resides on the aggregate from which the read/write fileset is moved, the command deletes the existing replication site and replica. It then creates a new replication site and replica on the aggregate to which the read/write fileset is moved, creating the new replica as a clone of the read/write fileset.
  - If the replica resides on the aggregate to which the read/write fileset is moved, the command has no immediate effect on the replica. However, the next time you issue the **fts release** command, the replica is changed from a full read-only copy to a clone of the read/write fileset.
  - If the replica resides on neither of the aggregates involved in the move, the command has no effect on the replica.

Recall that a clone fileset is a backup or read-only fileset at the same site (File Server machine and aggregate) as its read/write source fileset. Because a clone fileset shares data with its source fileset, it requires potentially much less space than a full replica on a different aggregate. (See Chapter 10, “Making Filesets and Aggregates Available” on page 147 for information about clone filesets and the sharing of data between different versions of the same fileset.)

Additional Considerations for Moving Filesets

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:

1. 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.
2. Fileset IDs are unique only for the cell in which they are assigned. Because of this, introducing a fileset into a different cell may cause conflict in the new cell between the fileset IDs of the newly introduced fileset and an existing fileset. A fileset ID conflict causes one of the two conflicting filesets to be inaccessible.

Note: These restrictions also prevent you from dumping and restoring a fileset between cells of the same name, even if you first remove the old cell and recreate a new cell with the same name. For the purposes of fileset movement, two cells are different, regardless of whether they share a common name.

Moving a Read/Write Fileset

To move the read/write version of a DCE Local File System fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** files on both the source and destination machines, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entries for the source machine, the destination machine, and any machines on which replicas of the fileset reside. In addition, the source machine (specified with the **-fromserver** option) must be listed in the **admin.ft** file on the destination machine (specified with the

-toserver option). If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

If the fileset is to be moved to a new File Server machine, and a read-only version of the fileset using Release Replication already resides on the destination machine, enter the **fts rmsite** command to remove the replication site from the destination machine.

```
$ fts rmsite -fileset {name | ID} -server machine -aggregate name
```

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 hostname, 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 hostname, 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}
```

Note: If you are moving a DCE Local File System fileset from an OS/390 system to an OS/390 system, it is recommended that you issue **fts move** from an OS/390 system.

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-Local File System fileset or any of the three types of DCE Local File System 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 Local File System filesets, you can also perform incremental dumps of only those files that have changed since a specified fileset version. Every DCE Local File System 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 Local File System 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 Local File System fileset.)

The **fts restore** command restores information from a previously dumped fileset back into the file system. Although you can dump a non-Local File System fileset or any of the three types of DCE Local File System 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 Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset. Similarly, a dump file of a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System 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.

Important Note to Users

The **fts dump** and **fts restore** commands are not available for OS/390 HFS filesets or RFS filesets in OS/390 DFS.

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 Local File System 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 Local File System 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 Local File System or non-Local File System) 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-Local File System fileset must exist before the non-Local File System partition on which it resides can be exported to the DCE namespace; therefore, when restoring a dump file as a non-Local File System fileset, you must use the **-overwrite** option to overwrite the existing non-Local File System 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 a another cell. (See "Moving DCE Local File System Filesets" on page 213 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 Local File System 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 Local File System fileset, omit the **-ftid** option to let the FL Server allocate a new ID number; if you are overwriting an existing DCE Local File System 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-Local File System fileset, do not use the **-ftid** option. Omit the option to continue to use the fileset ID number specified for the non-Local File System fileset in the entry for the partition on which the fileset resides in the **dfstab** file. The restored dump file overwrites all data on the non-Local File System 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.

Dumping a Fileset

To dump a fileset, perform the following steps:

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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) 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* or *mm/dd/yyyy 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"`).

Note: Valid values for *yy* are 00 to 37, which are interpreted as the years 2000-2037, and 70 to 99, which are interpreted as 1970-1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038-2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038. For a *yyyy* specification, valid values for *yyyy* are 1970-2069. However, values between 2038-2069 are reduced to 2038.

If specified, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The default time is 00:00 (12:00 a.m.).

If a value is entered for the year (*yy*) that falls between the range of 38 and 69, a random time is generated for the default time.

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 Local File System 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 Local File System fileset for which multiple replicas exist.

The **-file** *filename* option specifies the complete pathname of the file in which the dump is to be stored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is sent to standard output (**STDOUT**).

Restoring a Dump File to a New Fileset

To restore a dump file to a new fileset, perform the following steps:

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 READ (**r**) ACL permission for the dump file and the WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) 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 10, "Making Filesets and Aggregates Available" on page 147 for more information on fileset naming conventions.)

   The **-file** *filename* option specifies the complete pathname of the file that is to be restored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is taken from standard input (**STDIN**).

The **-ftid** *ID* option specifies the fileset ID number that is to be assigned to the fileset. If this option is omitted, a new ID number is allocated for the fileset. Use this option with great care; make sure the fileset ID number that you specify is not in use.

5. Issue the **fts crmount** command to create a mount point in the file system for the new fileset, making its contents visible to other users:

   ```
   $ fts crmount -dir directory_name -fileset {name | ID}
   ```

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

Restoring a Dump File by Overwriting an Existing Fileset

To restore a dump file by overwriting an existing fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to be stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be overwritten is stored. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Verify that you have the READ (**r**) 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 Local File System fileset; omit this option when restoring a dump file as a non-Local File System 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-Local File System fileset.

4. If read-only copies of the former read/write fileset exist, use the **fts update** command to replace them with replicas of the new fileset. If a backup version exists, use the **fts clone** command to replace it with a backup version of the new fileset.

Removing DCE Local File System Filesets

You use different commands to remove the different types of DCE Local File System filesets.

- To remove a read/write or backup DCE Local File System fileset, use the **fts delete** command.
- To remove a read-only DCE Local File System fileset, use the **fts rmsite** command; the **fts rmsite** command removes the specified replication site from the FLDB entry for the fileset and instructs the Replication Server at the site to remove the read-only filesets that reside there. You cannot use these commands to remove a non-Local File System fileset. (See "Removing Non-Local File System Filesets and Mount Point" on page 222 for information on removing non-Local File System filesets.)

The following list provides information about the effects of deleting the different types of filesets, certain restrictions that apply, and general guidelines you should follow:

Read/write filesets Before using the **fts delete** command to remove a read/write fileset that is replicated, you must first remove all replication sites and read-only replicas of the fileset; otherwise, the command fails. When you remove a read/write fileset for which a backup version exists, the backup version is automatically removed. As a result, removing a read/write fileset causes the entry for the fileset to be deleted from the FLDB. A backup fileset can exist after its read/write source is deleted if a delete operation is interrupted prior to completion.

When you remove a read/write fileset, you also need to remove its mount point with the **fts delmount** command so that users do not continue to try to access the fileset's contents. It is better to remove a fileset's mount point before deleting the fileset. Because the backup version is automatically removed at the same time, you also need to remove the backup version's mount point.

Read-only filesets You can use the **fts rmsite** command at any time to remove a fileset's replication site and the read-only replica that resides at the site. The information for the removed site is deleted from the entry for the fileset in the FLDB, but the fileset ID of the read-only version remains in the entry, so you can recreate the read-only version at any time. If you remove a replication site while other replication sites still exist for the fileset, the status flag for the read-only version in the FLDB entry remains unchanged; if you remove the last replication site for a fileset, the status flag for the read-only version is changed to `invalid`. (See Chapter 10, "Making Filesets and Aggregates Available" on page 147 for more information about using the **fts rmsite** command to remove a read-only fileset.)

Backup filesets The backup version of a fileset is automatically removed when the read/write version of the fileset is removed. However, you can use the **fts delete** command to remove a backup fileset at any time. To do so, append the **.backup** extension to the name of the fileset to be removed. Removing a backup fileset changes the status flag for the backup version to `invalid` in the FLDB entry for the fileset. The fileset ID of the backup version remains in the FLDB entry, so you can recreate the backup version at any time. When you remove a backup fileset, you should also remove the backup fileset's mount point.

Removing a fileset frees the space that fileset occupies on disk. When you remove a read-only fileset, other versions of the replica usually still exist (or the replica can be easily recreated). However, when you delete a read/write or backup 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 the fileset on tape.

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 "Other Commands for Removing Filesets" on page 221 for more information about these commands.)

Removing a DCE Local File System Fileset and Its Mount Point

To remove a DCE Local File System 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 WRITE (**w**), EXECUTE (**x**), and DELETE (**d**) 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 at the read/write fileset's site.
4. Issue the **fts delete** command to remove the read/write and backup versions of the fileset:

   ```
   $ fts delete rmsite -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. 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 each mount point that you want to remove.

   If you previously mounted the backup version as a subdirectory of the read/write fileset's root directory, removing the read/write version's mount point makes the backup version's mount point inaccessible. If you mounted the backup version at a separate directory, you must explicitly remove the backup version's mount point, again using the **fts delmount** command.

Other Commands for Removing Filesets

Under normal circumstances, always use the **fts delete** or **fts rmsite** command to remove a fileset. These commands automatically record the deletion in the FLDB. Under special circumstances, however, you may need to use the following commands. Keep in mind that if the FLDB and the filesets are consistent with each other, these commands make them inconsistent. Never use these commands unless absolutely necessary.

- Use the **fts delfldbentry** command to remove an FLDB entry that mentions a particular fileset. If versions of the fileset still exist at sites, they are not affected. This is useful if you are certain that a fileset removal was not recorded in the FLDB, and you do not want to use the **fts syncfldb** and **fts syncserv** commands to synchronize the entire FLDB. Use the **fts lsfldb** or **fts lsft** command to determine if a fileset removal was recorded in the FLDB.
- Use the **fts zap** command when it is urgent that a fileset be removed from its site, but the FLDB is inaccessible (for example, if the FL Server is unavailable). You can then remove the fileset's entry from the FLDB by entering the **fts delfldbentry** command or by entering the **fts syncfldb** and **fts syncserv** commands to synchronize the FLDB. The **fts zap** command, like the **fts delete** command,

cannot be used to remove a DCE Local File System fileset that is also mounted locally. Remove the fileset's local mount point before attempting to delete the fileset.

The following sections provide brief descriptions of the syntax and use of these commands.

Removing a Fileset's FLDB Entry Without Removing the Fileset: To remove a fileset's FLDB entry without removing the fileset, perform the following steps:

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.

Removing a DCE Local File System Fileset Without Updating Its FLDB Entry

To remove a DCE Local File System fileset without updating its FLDB entry, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset to be removed resides. If necessary, issue the **bos lsadmin** command to verify the members of the administrative list.
2. Enter the **fts zap** command to remove the fileset without recording the removal in the FLDB:

   ```
   $ fts zap -ftid ID -server machine -aggregate name
   ```

   The **-ftid** *ID* option specifies the fileset ID number of the fileset that is to be removed.

Removing Non-Local File System Filesets and Mount Point

When you remove a non-Local File System 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-Local File System fileset from the disk.)

To remove a non-Local File System 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-Local File System 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 **/opt/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).

Important Note to Users

In OS/390 DFS there is an **/opt/dfslocal/etc/ioepdcf** file that is used as the initialization file.

Any of these steps performed alone makes the fileset inaccessible. However, you should always perform all of the steps whenever you remove a non-Local File System fileset to prevent future problems, such as a mount point that references a fileset that is no longer exported.

To remove a non-Local File System 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 WRITE (**w**), EXECUTE (**x**), and DELETE (**d**) 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 each mount point that 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 partition in the **dfstab** file has the following format:

   ```
   /dev/ufs1 /usr  ufs  1  0,,18756
   ```

Locking and Unlocking FLDB Entries

The FL Server locks the FLDB entry for a DCE Local File System or non-Local File System 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; use the **fts unlock** command to unlock a specific FLDB entry.

The **fts unlockfldb** command unlocks a set of entries based on the options you provide.

When you :

- Provide no options, all locked entries are unlocked.
- Specify a File Server machine with the **-server** option, all locked entries with that machine in a site definition are unlocked.
- Specify an aggregate with the **-aggregate** option and a File Server machine with the **-server** option, all locked entries with that aggregate and that machine in a site definition are unlocked.

Determining Whether an FLDB Entry is Locked

To determine whether an FLDB entry is locked, issue the **fts lsfldb** command:

```
$ fts lsfldb -fileset {name | ID}
```

If the entry is locked, the word `Locked` appears on a line of the output of the command. The **fts lsft** command also displays the same line in its output if an entry it is examining is locked.

Locking an FLDB Entry

To lock an FLDB entry, perform the following steps:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be locked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts lock** command to lock the entry:

   ```
   $ fts lock -fileset {name | ID}
   ```

Unlocking a Single FLDB Entry

To unlock a single FLDB entry, perform the following steps:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be unlocked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts unlock** command to unlock the entry:

   ```
   $ fts unlock -fileset {name | ID}
   ```

Unlocking Multiple FLDB Entries

To unlock multiple FLDB entries, perform the following steps:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of any fileset to be unlocked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts unlockfldb** command to unlock all entries or only those entries on a specified server, on a specified aggregate, or on both:

   ```
   $ fts unlockfldb [-server machine] [-aggregate name]
   ```

Synchronizing the FLDB and Fileset Headers

In DFS, transparent file access is possible because the FLDB constantly tracks fileset locations. When the Cache Manager needs a file, it contacts the FL Server, which consults the FLDB to find the current location of the file. Therefore, the FLDB must accurately reflect the state of filesets on all File Server machines. To keep the FLDB accurate, all **fts** commands that affect fileset status automatically record the change in the appropriate FLDB entry.

Whenever you issue a command that changes the status of a fileset, the **fts** program 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.
- FL Server to set an intention flag in the FLDB entry to indicate the type of operation to be performed.
- 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`.
- 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 non-existent 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: 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 fileset header and the name found in FLDB entry, the fileset header is renamed to reflect the name in the FLDB entry. If the command encounters a fileset header marked as `Off-line`, but its FLDB entry lists it as being `valid`, the fileset header is placed `On-line`.

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.

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 Local File System 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 Local File System fileset is removed from the FLDB. You must use the **fts setrepinfo** and **fts addsite** commands to reconstruct the replication information.

Synchronizing Non-Local File System Filesets

The **fts syncfldb** and **fts syncserv** commands can be used to ensure the consistency of non-Local File System filesets. However, because non-Local File System 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-Local File System filesets and their FLDB entries.

For example, because non-Local File System filesets do not have fileset headers, the **fts syncfldb** command cannot determine the name of a non-Local File System fileset that has no FLDB entry. If the command determines that it needs to create an FLDB entry for a non-Local File System 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.

Important Note to Users

In OS/390 DFS, if you issue **fts syncfldb** or **fts syncserv** command for a non-Local File System fileset, the unique name that gets generated is *not* of the form **SYNCFLDB-ADDED-***number*. Instead, it uses the actual name of the OS/390 HFS data set in upper case, then the **fts rename** command can be used to rename it to its original name, if it was not originally named the OS/390 HFS data set name.

Similarly, because the **fts syncserv** command cannot destroy a disk partition, it cannot delete a non-Local File System fileset, even if it determines that the fileset needs to be deleted. Instead, the **fts** program displays an error message reporting the non-Local File System fileset that needs to be deleted to restore file system consistency. You must then enter the proper commands to remove the fileset from the DCE namespace.

Synchronizing Fileset Information

To synchronize the FLDB and fileset headers, perform the following steps:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on each machine that houses a version of any fileset stored at a site specified with the **fts syncfldb** or **fts syncserv** command. You must also be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine that houses a version of any fileset stored at a site specified with either command. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Issue the **fts syncfldb** command to make FLDB entries consistent with filesets that are stored at the specified site. *Repeat this command for every File Server machine in your cell.*

   ```
   $ fts syncfldb -server machine [-aggregate name]
   ```

   The **-aggregate** *name* option names the aggregate on the server machine specified with the **-server** option for which fileset headers and FLDB entries are to be compared. Do not use this option under normal circumstances; omitting it allows synchronization of all of the filesets on the machine specified with the **-server** option. Use it only when just a single aggregate needs to be synchronized.

3. Issue the **fts syncserv** command to make filesets that are stored at the specified site consistent with FLDB entries. *Repeat this command for every File Server machine in your cell.*

   ```
   $ fts syncserv -server machine [-aggregate name]
   ```

   The **-aggregate** *name* option names the aggregate on the server machine specified with the **-server** option for which fileset headers and FLDB entries are to be compared. Do not use this option under normal circumstances; omitting it allows synchronization of all of the filesets on the machine specified with the **-server** option. Use it only when just a single aggregate needs to be synchronized.

Verifying and Maintaining File System Consistency

Many operating systems use the **fsck** program to ensure file system consistency after a system failure. The **fsck** program checks the consistency of a file system and reports its findings. Optionally, it repairs problems that it finds in the file system. The **fsck** program (or its equivalent) is still used to return consistency to many types of non-Local File System partitions.

Important Note to Users

The **fsck** program is not available in OS/390 UNIX.

This section includes information on using the **salvage** command to repair a Logical Volume. In OS/390 DFS, **salvage** can be run as a batch job, as a TSO/E command, or from the OS/390 shell. The *xxx*.SIOESAMP(SALVAGE) member (where *xxx* is installation dependent) has sample JCL for running this program in batch.

For information on running **salvage** from TSO/E, or from the OS/390 shell, refer to “salvage” on page 747.

DFS employs a log mechanism and an additional system application, the DFS Salvager, to ensure the consistency of DCE Local File System aggregates. A log is kept of all changes made to metadata on a DCE Local File System 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 Local File System aggregate has its own log, which physically resides on the aggregate, where it is completely transparent to users.

The DFS Salvager returns consistency to a file system when the system is restarted by replaying the log. Under normal circumstances, replaying the log returns the file system to a consistent state. However, if the Salvager detects problems in the basic structure of the aggregate, if the log mechanism is damaged, or if the physical storage medium of the aggregate is suspect, replaying the log cannot restore consistency. In these cases, a system administrator must invoke the Salvager a second time to examine and repair the structure of the aggregate.

Overview of the DFS Salvager

The DFS Salvager is used to replay the log on an aggregate and, if necessary, to find and repair file system inconsistencies that cannot be repaired by replaying the log. Along with the normal consistency guarantees provided by the log mechanism, the Salvager performs the same type of functions as the **fsck** program in other operating systems; it reads the metadata that describes the contents of a file system, analyzes the internal organization and structure of the file system, and detects and repairs inconsistencies.

The Salvager is invoked with the **/opt/dfsglobal/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. The Salvager can also be used to verify an aggregate before recovery is run, but it typically finds problems with an unrecovered aggregate that it would not find were recovery run first.
- 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 Local File System 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.

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: 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.

The Salvager cannot be used 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: The Salvager also exits if it is run on an aggregate that houses a locally mounted fileset.

- The aggregate on which it is run contains a non-Local File System superblock whose creation time is more recent than that of its DCE Local File System superblock.

- The size of the aggregate that is recorded in its DCE Local File System superblock exceeds the capacity of the partition on which the aggregate resides.

At the prompt, the issuer can choose to cancel or continue the operation. If the operation is continued 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.

Attention: Never attempt to salvage an exported aggregate or an aggregate housing mounted filesets. In most cases, the filesets on an aggregate that requires salvaging are unmountable. However, you always need to verify that the filesets on an aggregate are not mounted and unmount them if necessary before attempting to salvage the aggregate. You also need to use the **dfsexport** command to detach the aggregate from the global namespace if is is currently exported.

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 READ (**r**) and WRITE (**w**) 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 READ (**r**) permission for the aggregate.
2. Ensure that the DCE Local File System 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 Local File System aggregate that is to be recovered, verified, or salvaged. The aggregate ID of the aggregate is not an acceptable value.

   The **-recoveronly** option directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate.

   The **-verifyonly** option directs the Salvager to verify the specified aggregate. The Salvager examines the structure of the aggregate to determine if it contains any inconsistencies, reporting any that it finds.

   The **-salvageonly** option directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate.

   The **-force** option executes the Salvager in noninteractive mode. By default, the Salvager prompts for confirmation before proceeding in certain situations. Use this option to direct the Salvager to proceed with all operations without asking whether it should continue.

   The **-verbose** option directs the Salvager to produce detailed information about the aggregate as it executes. The information is useful primarily for debugging purposes. Use this option alone or with any other combination of options.

Interpreting Salvager Output

The Salvager displays output on the screen. 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, it displays the following message if it is instructed to recover an aggregate:

```
Will run recovery on device
```

If the Salvager is used to recover an aggregate and the log on the aggregate does not need to be replayed, the Salvager displays no further output. 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.

Note: 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
```

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 Local File System 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 12. Using OS/390 Data Sets from DFS Clients

This chapter explains what you need to know when you use OS/390 data sets from a DFS client workstation. DFS for OS/390 can export OS/390 record files (sequential, PDS, PDSE, and Virtual Storage Access Method (VSAM)). This makes OS/390 record data available to DFS client applications. The data is presented as byte stream data.

Note: Generation Data Groups (GDG) are not supported.

Mapping Between the DFS Client's View and OS/390 Record Data

In OS/390, a file is called a data set. These two terms are used interchangeably.

The files for a computer system are organized into a file system. The AIX, OS/2, OS/390 UNIX, and other DFS client environments use a byte file system which is a hierarchy of directories that can contain other directories or files. OS/390 record data, however, uses a non-hierarchical file system in which groups of data sets are referred to by specifying a high-level qualifier.

The OS/390 high-level qualifier can include the first (leftmost) qualifier of data sets, or the first and second qualifiers, or the first, second, and third qualifiers, and so on. For example, `SMITH` is the high-level qualifier for the files named `SMITH.TEST.DATA` and `SMITH.PROJ7.SCHED`, while `SMITH.TEST` is the high-level qualifier of `SMITH.TEST.DATA` and `SMITH.TEST.DOCS`.

Note: Only Integrated Catalog Facility (ICF) cataloged data sets are supported by the DFS File Server. Tape data sets are not supported.

DFS exports a set of files with a specified prefix as a single fileset. All files with the specified prefix are shown as one level of hierarchy. If the data sets specified include partitioned data sets, a second level of hierarchy is shown.

Mounting OS/390 Data Sets onto the DFS Namespace

If the following OS/390 data sets exist:

```
USERA.B()      with members M1 and M2
USERA.B.C
USERA.B.C.D
USERA.X.Y()    with members M1 and M2
USERA.X.Y.Z
```

The following is an example of DFS server commands and operations to export an RFS aggregate that contains OS/390 data sets that begin with the prefix USERA.

1. Add an **rfs** aggregate to the **devtab** file.

   ```
   * RFS devices
   define_ufs 3 rfs
   USERA
   ```

2. Create a fileset location database entry for the **rfs** fileset.

   ```
   fts crfldbentry -ftname mvs.fsusera -server abc -aggrid 102
   ```

3. Add the **rfs** aggregate to the **dfstab** file using the fileset ID number (for the read-write fileset) returned by the **fts crfldbentry** command (for example: 0,,1718).

   ```
   /dev/ufs3         rfs3    ufs  102   0,,1718
   ```

4. Export the **rfs** fileset.

   ```
   dfsexport -aggregate rfs3
   ```

5. Create a DFS mount point for the **rfs** fileset.

   ```
   cd /.../abc.com/fs
   mkdir mvsfs
   cd mvsfs
   fts crmount -fileset mvs.fsusera -dir /:/mvsfs/mvsusera
   cd mvsusera
   ```

Figure 16 presents a view of the OS/390 files.

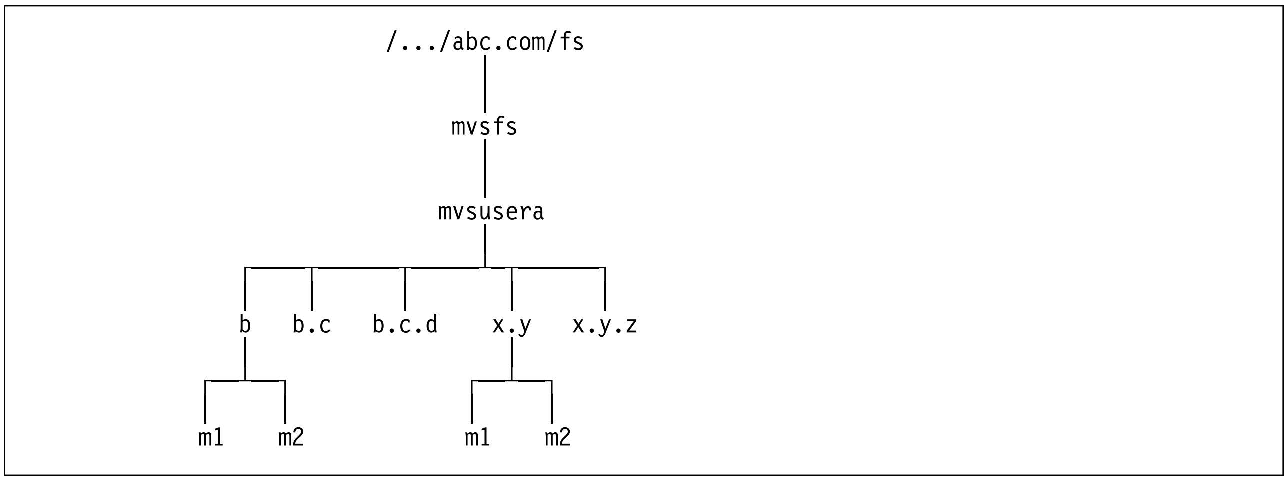


Figure 16. OS/390 Data Set Hierarchy

This allows you to define one level of directories under the mount point (**mvsusera**). Thus, you can issue **mkdir** to create a directory (stored as a PDS or PDSE) and then create files (stored as PDS or PDSE members) within that directory.

Alternatively, if the prefix specified in the **devtab** is an actual data set, DFS attempts to export the data set. If the data set is a PDS (or PDSE), it is handled as a directory and the members are exported. If it is any other type of data set, the export for that data set fails. (The name that is specified for export must be able to be handled as a directory.)

Reading, Writing, and Creating OS/390 Data Sets

You can use the DFS File Server to read data stored in an OS/390 data set from your DFS client system. You can also write data to an OS/390 data set (stored on OS/390 DASD) from your DFS client system. The OS/390 data sets appear as files in the DCE namespace on your DFS client system.

You can create OS/390 data sets from a client system using the DFS File Server. The default record data set creation attributes specified by the system administrator are used to create OS/390 data sets, unless the user overrides them. These attributes determine how the OS/390 data set is structured and where it is stored. You can override the data set creation attributes at file creation time.

Sharing Data

The sharing semantics that local OS/390 applications accessing record data expect and the semantics that DFS clients expect are very different. DFS clients expect multiple users to be able to open a file for write. The file integrity is protected through the use of advisory byte range locks. Local OS/390 applications, in general, do not expect other users to be writing to the file (data set) while their application is writing.

Also, DFS clients request tokens before reading or writing data. Possession of the token means that the client can act on the file without communicating to the DFS File Server. The DFS File Server keeps track of tokens it has given out. When a DFS client requests a token that conflicts with tokens that the DFS File Server has given out, the DFS File Server revokes (requests that they be given back to the server) those tokens. DFS clients that receive revokes always give back the tokens as soon as they can. DFS clients expect that all requestors are properly requesting and returning tokens. Local OS/390 applications using record data sets do not request tokens.

As a result, DFS must provide one set of semantics and token management to the DFS clients and another set of semantics with no token management to OS/390 record data applications. In order to provide this, DFS must construct a "wall" between the DFS clients and the OS/390 record data applications that are attempting to access the same OS/390 record file. DFS only allows either DFS clients or OS/390 record applications to write the data, but not both. In some cases, it does allow them both to read the data.

The general technique that DFS uses to accomplish this is to have the DFS File Server do a dynamic allocate on any record data set that a DFS client is attempting to access.

1. If the DFS client is attempting to read record data, and the data set is not a PDS, the DFS File Server attempts a DISP=SHR allocation for the data set.

    a. If the data set is free, then the dynamic allocate issued by the DFS File Server is successful and the DFS File Server provides the record data set to its DFS clients for reading.

       An OS/390 batch job that subsequently attempts to access that same record data set with an allocation of DISP=SHR will successfully access the data set.

       An OS/390 batch job that subsequently attempts to access that same record data set with an allocation of DISP=OLD will wait for the unallocate to occur. The DFS File Server is notified that an OS/390 batch job is waiting for the unallocate. (DFS uses the Event Notification Facility (ENF) and an event supported by Global Resource Serialization (GRS) for resource contention. See the *OS/390 MVS Programming: Authorized Assembler Services Guide*, GC28-1763, for information on the Event Notification Facility.) In this case, the DFS File Server frees up the record data set as soon as it can.

    b. If the data set is not free, the DFS File Server denies access to the data set. It appears to the DFS clients that the data set is unavailable. (DFS clients cannot wait an indefinite amount of time for the OS/390 batch job to complete.)

2. If the DFS client is attempting to write record data, or the data set is a PDS, the DFS File Server attempts a DISP=OLD allocation for the data set.

    a. If the data set is free, then the dynamic allocate issued by the DFS File Server will be successful and the DFS File Server provides the record data set to its DFS clients for writing.

       An OS/390 batch job that subsequently attempts to access that same record data set with an allocation of DISP=SHR or DISP=OLD will wait for the unallocate to occur. The DFS File Server is notified that an OS/390 batch job is waiting for the unallocate. In this case, the DFS File Server frees up the record data set as soon as it can.

b. If the data set is not free, the DFS File Server denies access to the data set. It appears to the DFS clients that the data set is unavailable. (DFS clients cannot wait an indefinite amount of time for the OS/390 batch job to complete.)

ISPF users attempting to access a record data set that has been accessed by DFS clients may be denied access because the DFS File Server may still have the data set allocated. ISPF (and TSO users) use dynamic allocation and this does not cause resource contention (and therefore no resource contention event occurs). The allocation request is simply denied. The DFS File Server does not recognize that the data set is being requested for use.

In this case, the user can submit a simple batch job that allocates the data set. This causes resource contention and therefore causes the DFS File Server to free up the data set as soon as it can. The batch job can be as simple as:

```
//JOBNAME JOB
//STEP1   EXEC   PGM=IEFBR14
//DD1     DD     DSN=USERA.B.C,DISP=OLD
```

Another technique can be used by DFS client users with the proper authority to force the DFS File Server to free up an OS/390 data set. A user with RACF ALTER authority to the OS/390 data set can issue the **touch "*name*,release"** command from a DFS client machine. For example, **touch "b.c,release"** would cause the DFS File Server to free up the **USERA.B.C** data set as soon as it can. (If *name* is a member of a PDS or PDSE data set, the entire PDS or PDSE data set is released.)

A second form of the command causes a release to occur for every object in the directory that the DFS File Server has allocated. When issued at the mount point of an RFS fileset, it causes all OS/390 data sets in the RFS fileset to be freed. The second form is issued as the **touch "*,release"** command. This could be used if the administrator has made RACF changes to the RFS fileset high-level qualifier. It will cause those changes to take effect without requiring the unexport/export of the RFS fileset.

When you issue either form of the touch release command, a touch error message is expected. If the release was successful, the message indicates that the system attempted to create the file but could not because the object already exists. For example,

```
/:/mvsfs/mvsusera -> touch "*,release"
touch: 0652-046 Cannot create *,release
```

This is because the DFS client sends this request as a create file request (since the file "*,release" does not exist). Since we are not really creating this file at the server (rather, we are doing release processing against a file or set of files), we must send an error code (in this case EEXIST - object already exists) back to the DFS client when the release processing is completed successfully to keep the DFS client from continuing with its create file processing. (It would request the attributes of the new file, etc.)

If the user does not have RACF ALTER authority to the file(s), the DFS server will send back EACCES (the user does not have the required permission) to the DFS client.

An operator force command (**modify dfs**) is provided to allow the operator to force the DFS File Server to free up an OS/390 data set if an important batch job has been waiting too long.

Forcing a Data Set to be Freed by DFS

The following operator command may be issued to force the DFS Server to free a data set for access by a batch job:

modify dfs,send dfskern,release,*data-set-name*

where *data-set-name* is the name of the data set that the DFS Server should make available.

Refreshing RFS File Names

DFS uses the Event Notification Facility to determine when an OS/390 batch job is attempting to allocate a data set that is currently being accessed by DFS clients. When the DFS Server detects this, it revokes all tokens and unallocates the data set (thus making it available to the OS/390 batch job).

Since there is no event to indicate when data sets are created, deleted, or renamed by an OS/390 batch job, the DFS Server refreshes its cache of exported data set names and attributes at a regular interval. This interval will be controlled by the **_IOE_RFS_STATUS_REFRESH_TIME** environment variable in the **dfskern** process. The time is specified in seconds and the default is 600 (ten minutes). In this case, a DFS client listing file names when positioned at an **rfs** root directory may not see a new file created by an OS/390 job for up to ten minutes after it is created.

Note: This only affects commands that list file names and attributes. A newly created file can be directly referenced immediately after it is created.

Special OS/390 Considerations for Record Data

In addition to mapping between the DFS client's view and OS/390 file systems, you should be aware of the other ways in which the OS/390 record data might differ from hierarchical byte data. These differences include:

- Selecting an OS/390 data storage format
- File size determination and time stamps
- Ownership and permissions
- Client caching
- OS/390 record file names.

Selecting an OS/390 Data Storage Format for Record Data

DFS clients can access OS/390 data sets. These OS/390 data sets are record-oriented and can be sequential, direct, VSAM, partitioned, and so forth, and also can contain variable or fixed-length records. AIX, OS/2, OS/390 UNIX, and other DFS client environments, however, are byte-oriented and may write or read at certain byte offsets in the file.

The DFS File Server can map DFS client requests to most types of OS/390 data set organizations. However, how the time stamps and file size value are handled depends on the type of OS/390 data set used, and the file size processing can affect performance.

Direct reads with the data set attributes **recfm(fbs)** or **recfm(f)** can be fast because in some cases, the DFS File Server can determine the physical block addresses from the record offsets. The OS/390 sequential file organization with **recfm(f)** or **recfm(fbs)** on DASD allows for efficient updating or reading at any offset in the file. Other supported OS/390 access methods (for example, VSAM) may not perform as efficiently.

File Size Determination and Time Stamps

The DFS File Server determines how to handle the file size value and time stamps depending on the type of OS/390 data set used and the attributes used to access the data set. See “Handling of the File Size Value” on page 246 for more information of file size determination and “Handling of the Time Stamps” on page 248 for more information on handling time stamps.

Ownership and Permissions

In DFS client environments, files have ownership and permissions stored as part of the file. These can be displayed at the DFS client (e.g., **ls -al**). OS/390 record files are not, in general, "owned" in this same sense. A Dummy UID and GID are displayed for RFS files. The UID displayed is controlled by the processing attribute in the attributes file. If **setownerroot** is specified, **UID = 0** (root) is displayed. This is the default. If **setownernobody** is specified, **UID = -2** (nobody) is displayed. **GID = 0** (system) is always displayed. Users can issue **chown** to change the owner in the DFS File Server memory but this is temporary and has no effect on the authorization users have for files or directories.

Just as it is not necessarily possible to determine a user's authorization to a DCE Local File System file by looking at the UID, GID, and permissions (due to permission extensions by ACLs), it is also not necessarily possible to determine a user's authorization to an OS/390 record file by looking at the UID, GID, and permissions displayed at the DFS client. Permission checking is done by RACF. Access to a data set is granted, provided that the OS/390 security subsystem allows access to the data set. Dummy permissions will be presented for OS/390 record files. DFS always shows the permissions as 777 (rwxrwxrwx) for directories (PDS or PDSE).

For files, DFS shows 777 (rwxrwxrwx) if **executebiton** is specified in the attributes file or 666 (rw-rw-rw-) if **executebitoff** is specified. Users can issue **chmod** to change the permissions in the DFS File Server memory but this is temporary and has no affect on the authorization users have for files or directories.

DFS Client Caching

OS/390 record file data is cached at DFS clients in the normal manner. DFS clients will request and return tokens as usual. However, if the DFS File Server fails and restarts, and you are currently positioned with an RFS file system, then you will need to change directory (**cd**) back to the root of that RFS file system before you can access files within that RFS file system.

OS/390 Record File Names

As explained in “Mapping Between the DFS Client's View and OS/390 Record Data” on page 235, OS/390 record file names consist of segments separated by a dot with a maximum length of 44 characters. Each segment can be from one to eight alphanumeric characters. (See the *DFSMS/MVS Version 1 Release 3, Using Data Sets* book, SC26-4922 for information on data set naming.) Each OS/390 record file appears as a byte file to DFS clients. A PDS appears as a directory with the members appearing as files within the directory. PDS member names are limited to eight characters.

OS/390 data set names are always in upper case characters. This means that file names are displayed to DFS client users in upper case and need to be entered in upper case. A file name that was entered in lower case is not found or is not created.

There is, however, a processing attribute in the attributes file (see “Attributes File (rfstab)” on page 341) called **maplower**. When this attribute is specified in the attributes file, users can enter lower case letters for file names and they are mapped to upper case by the DFS File Server. Also, when the (upper case) file names are displayed to the user, they are mapped to lower case by the DFS File Server. **maplower** is the default.

You are cautioned, however, that when the **maplower** processing attribute is in effect, you should only use lower case letters for file names. If you create a file with the name `AbCd`, it becomes an OS/390 data set named `ABCD`. When the file name is displayed to the (DFS client) user, it appears as `abcd`. The file name would be displayed as a different name than the user entered when it was created.

Also, many byte file systems typically allow file names to begin with a dot. This is an invalid name for an OS/390 data set. However, a processing attribute in the attributes file called **mapleaddot** causes a leading dot in a file name to be mapped to a dollar sign in the OS/390 data set name and back to a dot for the DFS client. **mapleaddot** is the default.

Creating OS/390 Files

This section describes how to create the various types of data sets (files) supported by the DFS Server.

The examples shown are for the AIX RS/6000® platform. Any examples for other platforms have been indicated.

Overriding Data Set Creation Attributes

When you create an OS/390 file, default file creation attributes are applied, unless you override them. The attributes are passed to the OS/390 host.

Data set creation attributes are controlled in the following ways, in increasing order of priority:

- Default server data set creation attributes (see “Attributes File (rfstab)” on page 341)
- Default installation data set creation attributes, specified by the system administrator in the attributes file (see “Attributes File (rfstab)” on page 341)
- DFSMS™ data class attributes
- Data set creation attributes specified for the fileset in the **devtab** file (see “devtab” on page 368)
- Data set creation attributes specified at file creation, for example, in the **mkdir**, **vi** (edit), or **cp** (copy) commands (highest priority).

Note: These data class attributes are not supported by the DFS Server:

- Retention period/Expiration date
- Number of volumes
- VSAM imbed index option
- VSAM replicate index option
- CI size of data component
- Percentage of CI or CA free space.

Preparing to Create an OS/390 File

When you create an OS/390 file, you may want to specify what type of file to create.

These following types of files are supported by the DFS Server:

- Physical sequential (PS)
- Direct access (DA)
- Partitioned data sets (PDS)
- Partitioned data sets extended (PDSE)
- VSAM KSDS
- VSAM ESDS
- VSAM RRDS
- Sequential Access Method (SAM) extended format data sets.

Note: Keyed access to files is not supported by DFS clients.

Naming OS/390 Files: When naming conventional OS/390 files, you must follow the OS/390 file naming conventions, as described in the *DFSMS/MVS Version 1 Release 3, Using Data Sets* book, SC26-4922.

An OS/390 file name (or data set name) can consist of one or several simple names joined so that each represents a level of qualification. For example, the OS/390 file name `DEPT58.SMITH.DATA3` is composed of three qualifiers.

The following characteristics apply to the OS/390 file name:

- Each qualifier consists of 1 to 8 alphanumeric characters, national characters (@, #, $), or a hyphen (-).
- Each qualifier must start with an alphabetical or national character.
- The period (.) separates simple names from each other.
- Including all simple names and periods, the length of the OS/390 file name must not exceed 44 characters. Note that the OS/390 high-level qualifier that was exported (in the previous example, USERA) is added as a prefix to the file name. So, if you created file **/.../abc.com/fs/mvsfs/mvsusera/b.c.d.e**, DFS would create OS/390 dataset **USERA.B.C.D.E**.
- PDS and PDSE member names can be up to 8 characters long.

Restrictions Using Alias Names for OS/390 Files: For PDSs and PDSEs, alias names (for member names) are not supported. They are not displayed when you list the names in a directory (that is really a PDS or PDSE). You cannot access a file (member) by its alias name. If you try to create a file in the directory that has the same name as an alias, it will be denied.

Creating Physical Sequential Files

Note that the following examples assume that an **rfs** fileset is mounted at **/.../abc.com/fs/mvsfs/smith** and the data set prefix in **devtab** is SMITH. In addition, it is assumed that **abc.com** is your local cell so that **/:** is equivalent to **/.../abc.com/fs**. Finally, it is assumed that these files are translated between ASCII and EBCDIC characters by an administrator specification of:

- **_IOE_RFS_TRANSLATION=ON**, or
- Text in the **devtab** entry for **rfs** fileset, or
- Text in the attributes file for the RFS fileset.

When creating a physical sequential (PS) file, specify the **dsorg(ps)** attribute (if it is not the default already) with a file creation command, such as the **vi** UNIX (or AIX) command.

```
$ vi "/:/mvsfs/smith/new,dsorg(ps)"
```

When you save the file using **vi**, you have just created a new OS/390 PS file named `SMITH.NEW`.

When reading or changing data in a Physical Sequential file with fixed-length records in text mode, the **blankstrip** processing attribute in the attributes file controls how trailing blanks are handled. If **blankstrip** is specified in the attributes file, trailing blanks are removed from the end of each record when it is read, and blanks are padded to the end of each record when it is written. **blankstrip** is the default. If **noblankstrip** is specified in the attributes file, trailing blanks are not removed from the end of each record when it is read, and blanks are not padded to the end of each record when it is written. In this case, each record written must be the correct size or an I/O error is reported.

Note: When copying a file to an RFS fileset, if you are overriding the data set creation attributes, you must specify the target file name in the **copy** command. For example:

```
cp file1 "/:/mvsfs/mvsusera/newfile1,space(2,5),cyls"
```

The previous example is correct.

```
cp file1 "/:/mvsfs/mvsusera,space(2,5),cyls"
```

The previous example is incorrect.

Creating Direct Access Files

When creating a direct access (DA) file, specify the **dsorg(da)** attribute (if it is not the default already) with a file creation command, such as the **vi** UNIX (or AIX) command.

```
$ vi "/:/mvsfs/smith/new,dsorg(da)"
```

You have just created a new OS/390 DA file named `SMITH.NEW`.

Creating PDSs and PDSEs

Partitioned data sets (PDSs) and partitioned data sets extended (PDSEs) can be used as directories, and their members are files within those directories. An illustration of the use of PDSs to act as directories is shown in Figure 16 on page 236. For general information on PDSs and PDSEs, see the *DFSMS/MVS Version 1 Release 3, Using Data Sets* book, SC26-4922.

You cannot create new directories within a PDS or PDSE, due to the nature of these data structures.

Creating a PDS or PDSE — mkdir dsntype(pds), dsntype(library): To create a PDS or PDSE, perform the following steps:

1. If creating a PDSE, use the UNIX **mkdir** (make directory) command specifying the **dsntype(library)** attribute to create a PDSE named `smith.datalib`:

   ```
   $ mkdir /:/mvsfs/smith/"datalib,dsntype(library)"
   ```

   If creating a PDS, use the UNIX **mkdir** (make directory) command specifying the **dsntype(pds)** attribute as follows:

   ```
   $ mkdir /:/mvsfs/smith/"datalib,dsntype(pds),dir(20)"
   ```

   Note: You can omit specifying the **dsntype(pds)** attribute, if **pds** has been specified for the **dsntype** attribute in the attributes file (see page 342).

2. You can use the UNIX **vi** command to create a PDS or PDSE member named `smith.datalib(member1)`:

   ```
   $ vi "/:/mvsfs/smith/datalib/member1"
   ```

   Input your text, save it, and quit.

You have now created a PDS or PDSE member. You can use the UNIX **cat** command to view the contents of your PDS or PDSE member.

Note: DFS Server supports a maximum of 14,562 members in a PDS or PDSE data set. When a DFS read-directory request on a PDS or PDSE is processed, the DFS Server will return up to 14,562 member names. Other requests, such as read and write, to individual members are not affected.

Removing a PDS or PDSE — rm, rmdir: To remove a PDS or PDSE, first make sure that the PDS or PDSE is empty. You can delete all members under the directory using the UNIX **rm** command. Then use the UNIX **rmdir** (remove directory) command. This example removes the `datalib` directory, and confirms its removal by a failed try to query it (**ls** is the UNIX list files command):

```
$ ls -alF /:/mvsfs/smith/datalib
drwxr-xr-x   2  100       110         0 Nov 20 14:48 ./
drwxr-xr-x  10  100       110         0 Nov 20 14:48 ../
-rwxr--r--   1  100       110       236 Nov 20 12:15 data1*
-rwxr--r--   1  100       110     18273 Nov 20 14:48 data2*
-rwxr--r--   1  100       110      8734 Nov 20 15:03 data3*
$ rm /:/mvsfs/smith/datalib/*
$ rmdir /:/mvsfs/smith/datalib
$ ls -alF /:/mvsfs/smith/datalib
ls: FSUM6785 File or directory "/:/mvsfs/smith/datalib" is not found
```

Accessing PDS or PDSE Members: There is more than one way to access PDS and PDSE members. For example, you could display the existing PDS member `smith.source(bigblue)` by entering either of these command sequences:

```
$ cat /:/mvsfs/smith/source/bigblue
```

or

```
$ cd /:/mvsfs/smith/source
$ cat bigblue
```

These two approaches are equivalent.

Updating or Extending a PDS or PDSE Member: The DFS Server does not generally support updating or extending a PDS or PDSE member directly. To update or extend a PDS or PDSE member, a client program must follow these steps:

1. Copy the file to the client machine
2. Update or extend the copied version on the local system
3. Truncate the original OS/390 file to zero size by sending a SETATTR request with zero file size
4. Copy the updated version on the local host to OS/390 by writing request

Some client editors follow the above steps, for example, OS/2 EPM and KEDIT editors, and the AIX and UNIX vi editor. Other editors do not follow the above steps, for example, OS/2 E editor and the OS/390 OEDIT editor. In the latter case the user must save the updated version into a new file.

When reading or changing data in a PDS member or PDSE member with fixed-length records in text mode, the **blankstrip** processing attribute in the attributes file controls how trailing blanks are handled. If **blankstrip** is specified in the attributes file, trailing blanks are removed from the end of each record when it is read, and blanks are padded to the end of each record when it is written. **blankstrip** is the default. If **noblankstrip** is specified in the attributes file, trailing blanks are not removed from the end of each record when it is read, and blanks are not padded to the end of each record when it is written. In this case, each record written must be the correct size or an I/O error is reported.

Renaming or Moving a PDS or PDSE Member: Record file system (RFS) files and directories can only be renamed or moved in a way that does not cause them to be moved from one directory to another.

If you are writing to a PDS or PDSE member and a timeout occurs, the timeout causes the member to close. The remaining write requests appear to append to a PDS or PDSE member. This operation is not supported and causes an I/O error. To avoid timing out, increase the timeout setting.

Wildcard Copy to a PDS or PDSE: To ensure that a wildcard copy, `cp /home/smith/* /:/mvsfs/smith/source`, to a PDS or PDSE can be completed successfully, a prior PDS member is closed and dequeued (if necessary) to allow the creation of a new member.

Limitations of Writing to a PDS: The PDS support in DFS adheres to the conventions used in OS/390. For example, you cannot have more than one member of a PDS open for writing at a time. If you try to write to a member of a PDS while another member is open for write by a different user, you get a “Permission denied” message.

A PDS member stays open for the timeout period specified in the appropriate timeout processing attribute, **filetimeout**, or until you try to create or write to another member.

Concurrent Writes to a PDSE: DFS supports concurrent writes to a PDSE. If you are writing to one member of a PDSE, another client can write to any other member in the same PDSE.

Creating VSAM Files

DFS supports three types of VSAM files: key-sequenced (KSDS), entry-sequenced (ESDS), and relative record (RRDS). However, keyed access and relative-number access to the files are not supported.

If you plan to update a VSAM data set (for example, with the **vi** editor or with the **cp** copy command), the data set must have been defined with the REUSE option. Trying to write back a VSAM data set that was not defined as reusable results in an “I/O error,” “failure to open,” or similar error message. When you create a VSAM file through the DFS server, the REUSE option is always specified by the server.

For more information on VSAM files, see the *DFSMS/MVS Version 1 Release 3, Using Data Sets* book, SC26-4922.

In the following example, the attributes indicate that:

- Spanned records are allowed
- Organization is key-sequenced
- Keys are 8 bytes long and start in position 0 of each record
- Average record size is 1024
- Maximum record size is 4096
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT.

```
$ cp ksds.old "ksds.new2,spanned,dsorg(indexed),keys(8,0),
   recordsize(1K,4K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

In the following example for creating a VSAM ESDS file, the attributes indicate that:

- Spanned records are allowed
- Organization is entry-sequenced
- Average record size is 1024
- Maximum record size is 4096
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT.

```
$ cp esds.old "esds.new3,spanned,dsorg(nonindexed),
   recordsize(1K,4K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

In the following example, the attributes indicate that:

- Spanned records are not allowed
- Organization is relative record, numbered in ascending order
- Average record size is 1024
- Maximum record size is 1024
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT.

```
$ cp rrds.old "rrds.new4,nonspanned,dsorg(numbered),
   recordsize(1K,1K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

Specifying Attributes Multiple Times

Specifying an attribute several times on a line does not cause an error. The line is read from left to right, and the last of any duplicate attribute is used. For example:

```
$ vi "file,recfm(vb),recfm(fb)"
```

This results in a file created with a fixed-blocked format.

Exploiting SAM Striped Files

With SAM striping, data I/O is done in parallel to improve performance. For a file with 16 stripes, data is processed on the first track of the allocated space on the first volume (that is, the first stripe), then on the first track of the second volume, and so on for all 16 volumes. Then, processing continues with the second track of all the volumes, then the third, and so on.

The DFS Server can support data set striping through the use of data class and storage class attributes that define extended format data sets. The DFS Server can exploit the performance of extended format data sets by reading multiple blocks at a time when reading ahead.

For more information on striped files, see the *DFSMS/MVS Version 1 Release 3, Using Data Sets* book, SC26-4922.

Handling of the File Size Value

Many file system commands (such as: **ls -al**) require the file size to be returned. This appendix explains some performance and accuracy considerations in obtaining the file size value.

The meaning of the file size value returned by the DFS file server and how fast the file size is returned depends on:

- Whether you use **text** or **binary** processing mode
- The type of OS/390 data set being accessed
- If the data set is system-managed
- Whether you use **fastfilesize** or **nofastfilesize** processing.

Storage of the File Size Value

Whether or not the file size value is already stored on your OS/390 system, affects how quickly files are accessed and depends on the type of OS/390 data set used.

System-Managed PS, VSAM, and PDSE Data Sets: For system-managed data sets, text and binary file size are saved on non-volatile storage (DASD) and maintained by the DFS File Server for the following data set types:

- Physical sequential (including striped)
- VSAM ESDS
- VSAM KSDS
- VSAM RRDS
- PDSE members.

When the DFS file server accesses a data set for the first time, it performs a read-for-size to get the text or binary file size and stores this value on DASD. Subsequent file size requests from clients do not cause the server to read for size, thus improving performance. However, when the data set is modified outside the server by a non-DFS application (for example, by the TSO editor), the stored file size could be incorrect. When the data set is accessed again by the server, read-for size is done to determine the correct file size.

Migrated System-Managed Data Sets Under DFSMS/MVS V1R3: DFSMS/MVS™ Version 1 Release 3 allows data set attribute accessibility for SMS managed data sets, without having to recall the data set if the data set is migrated under DFSMS/MVS V1R3. Supported SMS managed data set types:

- PS
- VSAM ESDS
- VSAM KSDS
- VSAM RRDS
- PDS
- PDSE.

Migrated PDS/PDSE members are not supported.

The DFS file server is able to obtain the attributes of a supported SMS managed migrated data set without recalling the data set. Attributes such as the record format and file size are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS managed migrated data set, thus improving performance. However, when the data set is modified outside the server by a non-DFS application (for example, by the TSO editor) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the server, a recall is done to determine the correct file size.

Non-System Managed, PDS, and DA Data Sets: The file size value for non-system managed data sets, PDS members, and DA data sets is cached in virtual storage until released but not written to DASD. Therefore, for these types of OS/390 data sets, the file size value is re-generated after the file is released or after the server is restarted.

How the File Size Value Is Generated

When a file is first accessed (for example with **ls -l** or **dir**), usually the entire file is read to determine its size, except for when specifying the **recfm(f)** or **recfm(fbs)** attributes where the binary size can be computed without reading the file. If the file is a system-managed PS, VSAM, or PDSE member, both binary and text file sizes are stored on DASD, so that subsequent file size requests do not require the file to be read.

Binary file size can be quickly generated by using **recfm(f)** or **recfm(fbs)** to specify a fixed-length record format for the OS/390 data set. With this format type, the server pads the last logical record with binary zeros in **binary** mode processing, because OS/390 always expects complete logical records. If the application tolerates these zeros, using **recfm(f)** or **recfm(fbs)** allows the binary size to be computed quickly because the number of bytes can be computed from the number of blocks, which is stored by OS/390.

If you need the exact file size and are using **binary** mode processing, map it to a variable-format, sequential data set on DASD so that the DFS file server does not need to pad a partially filled last OS/390 logical record to a record boundary.

For reading small files or the beginning of files, the read-for-size might not add any processing time. As the file is being read for size, the beginning of the file is stored in the buffers set aside by the **maxrdforszleft** site attribute, until the buffers are full. When the application reads the beginning of the file, this read is fast because it reads directly from the buffer.

OS/390 stores the number of blocks (rather than the number of bytes) in an OS/390 file. For most files, therefore, without reading the entire file, the DFS file server can only give an estimate of the number of bytes in the file, not the exact number of bytes in the file.

Using fastfilesize to Avoid Read-for-Size: If you can use an approximate file size for a PDS, PDSE, DA, or non-system managed data set, you can specify the **fastfilesize** attribute to improve performance. With this attribute, the server estimates the size without opening and reading the entire file.

PDS members	For PDS and PDSE members, the **fastfilesize** attribute gets the file size from ISPF statistics if they exist; otherwise, the size of the entire PDS or PDSE data set is returned as the member size.
PS or DA data sets	For PS or DA data sets, an approximate file size is calculated based on the device characteristics, the number of disk tracks in use, and the block size of the data set.
VSAM	For non-system-managed VSAM data sets, the estimated size using **fastfilesize** is the size of the data set.

The **fastfilesize** attribute speeds up data set access by calculating approximate file sizes during data set access. Use this only when you are browsing through files (using the **ls** UNIX command or the **type** OS/2 command, for example) because some commands (such as **cp** or **copy**) might not work correctly if **fastfilesize** is set. When modifying or copying a data set, the **nofastfilesize** attribute should be used to ensure accurate results.

nofastfilesize: When you use the default, **nofastfilesize** attribute, the DFS file server reads the entire file or member to get the file size. It stores the file size value in cache until release. Using this attribute might cause a delay when first accessing very large data sets.

Handling of the Time Stamps

UNIX file attributes define the following time stamps:

atime	The last time the file was accessed (read)
mtime	The last time the file was modified (write)
ctime	The last time the file status was changed (chmod)

The DFS file server handles time stamps differently for these types of data sets:

- System-managed PS data sets and system-managed VSAM data sets
- Direct Access data sets and non-system managed PS data sets
- Non-system managed VSAM data sets
- PDS and PDSE members.

Time Stamps for System-Managed VSAM and PS Data Sets

For system-managed PS data sets and system-managed VSAM data sets, *atime* and *mtime* are fully maintained, and the *ctime* is set to the *mtime*.

Time Stamps for Non-System Managed PS and DA Data Sets

For non-system managed PS and DA data sets, consider the following:

- How time stamps are stored
- The requirements of your workstation programs
- The type of OS/390 data set used to store the file.

Storing Time Stamps: For non-system managed PS and DA data sets, the DFS file server temporarily stores the time stamps in virtual storage, but not on DASD. These cached attributes are purged when the file is released or when the server is restarted. When the file is accessed again, the time stamps are re-generated.

Client Program Requirements: Some workstation-based utilities (such as **make**) rely on date and time stamps to determine whether to recompile. For example, **make** checks the update time of the object file with the source file and recompiles if the source has been updated. Before storing these types of files using the OS/390 server, examine them before moving them to ensure that these attributes are unimportant. In an environment which relies on such utilities, use HFS or DCE Local File System.

Generating Time Stamps: This is how the DFS file server generates *atime* and *mtime* for non-system managed PS and DA data sets from the OS/390 dates:

```
atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

time_increment is either the server local time or 23:59 hours. If *reference_date* or *creation_date* is equal to the server local date, the server local time is added. Otherwise, a fixed value of 23 hours and 59 minutes is added.

If *reference_date* = 0 (that is, the file has not yet been referenced), *atime* and *mtime* are set equal to *ctime*.

Time Stamps for Non-System Managed VSAM Data Sets

The time stamps for these types of data sets are set to the current time.

Time Stamps for PDSs and PDSEs

An OS/390 PDS data set can act as a UNIX directory. Members of the PDS are files within the UNIX directory. When the directory is accessed by the client, the UNIX times are expected for each file.

Ordinarily, OS/390 does not maintain time stamps for members of a PDS. The UNIX time stamps here are generated from the OS/390 creation and reference dates of the PDS data set containing the members. This is how the time stamps for PDS members are generated:

```
atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

ISPF is an OS/390 base element that does maintain some additional statistics for each member. They include the creation date and the last modification date and time.

If the ISPF time stamps are present for a PDS member, this is how the server generates the time stamps and initializes the UNIX times:

```
atime = mtime = modification_date + modification_time
ctime = ISPF_creation_date + time_increment
```

time_increment is either server local time or 23:59 hours as described for non-system managed PS and DA data sets.

The server also creates new ISPF statistics for PDS members created by the clients. The ISPF statistics are created even if existing members do not have statistics.

The time stamp information is saved in the PDS directory according to ISPF conventions. If STATS=ON was specified when the member was created, the server uses them to get more accurate attributes. Even if STATS=ON was not specified originally, the server writes back new time stamp information if the member is modified from the workstation.

Time stamp generation for a PDSE member is identical to that of a PDS with one exception. Accurate *mtime* of a PDSE member is returned to a client as the result of a file attribute request for that PDSE member.

Setting Time Stamps

DFS clients can issue commands that result in SETATTR requests (such as, touch) to set the *atime* and *mtime* for a system-managed PS or VSAM data set. For PDSE members, setting *mtime* is allowed, but setting *atime* is not supported. PDSE member *mtime* is also maintained by PDSE access methods, so it is modified when a TSO user modifies the PDSE member.

Chapter 13. Configuring the Cache Manager

This section describes the configuration of an OS/390 system to allow OMVS users and applications to act as DFS clients. The Cache Manager enables an OMVS user or application to access directories and files in the DFS global namespace. You can control several aspects of Cache Manager and client performance by configuring the Cache Manager as described in this section.

Important Notes to Users

- The terms **Cache Manager** and **DFS Client** are equivalent.
- Any statements that specify the user must login as **root**, refers to logging in with a user ID which has a **UID = 0**.
- The OS/390 system the DFS client (also known as the Cache Manager) is recognized as a physical file system running as a colony address space. Refer to the *OS/390 UNIX System Services File System Interface Reference*, SC28-1909, for more information on physical file systems and colony address spaces.

In OS/390, the Cache Manager address space **DFSCM** is started by OMVS as a result of the **BPXPRM***xx* **FILESYSTYPE TYPE(DFSC)** entry. Refer to "Cache Manager Initialization Parameters" on page 273 for a description of how to specify parameters used during OS/390 DFS client (Cache Manager) initialization.

The **DFSCM** address space is where the DFS client **ioecmini**, **ioedfsd**, and **ioelogin** processes run.

The **ioecmini** is specific to OS/390. It is called by OMVS to perform Cache Manager initialization functions and program calls by OS/390 UNIX in *cross memory mode* to service DCE Global (DFS) name space access requests. The **ioecmini** process queues works for the **ioedfsd** process threads to perform the requested work.

The **ioedfsd** process resolves CDS pathnames for the Cache Manager and accesses user authentication information necessary for effective communications with server machines. It communicates with DFS server machines using DCE RPC to read and write directory and file data defined in the DFS Global namespace.

Important Note to Users

The **ioedfsd** process is also unique in OS/390 in that it combines the functions of the traditional Cache Manager **dfsd** and **dfsbind** processes. In OS/390, references to the **dfsd** or **dfsbind** process should be taken to mean the **ioedfsd** process.

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 (an area of local disk or memory designated for temporary storage). Once the data is cached locally, the Cache Manager can access it as quickly as it can a local file.

The Cache Manager verifies that its cached files match the central copies at File Server machines by keeping the tokens that the File Exporters send with the files. A token acts as the File Exporter's promise to contact the Cache Manager if the centrally stored copy of a file changes while the Cache Manager has a cached copy. If the central copy changes, the File Exporter revokes the token; the Cache Manager sees that the token is revoked and retrieves the new version of the file when an application program next requests data from it.

Cache Manager Processes

Important Note to Users

In OS/390, any DFS Cache Manager related reference to the *kernel* should be taken to mean a colony address space.

The **ioedfsd** process controls the Cache Manager where **DFSCM** is recognized as a physical file system running as a colony address space.

In OS/390, you can customize and control several Cache Manager features by the following:

- Modifying the local HFS **/opt/dcelocal/etc/CacheInfo** file; or
- Adding Cache Manager initialization parameters to the **_IOE_CM_PARMS** parameter in the **/opt/dfslocal/home/dfscm/envar** file.

Important Note to Users

In OS/390, the DFS Cache Manager is invoked when OMVS is started or it can be stopped and restarted. Cache Manager customization and control parameters can be specified in the local HFS **/opt/dcelocal/etc/CacheInfo** file or in the **/opt/dfslocal/home/dfscm/envar** file. The parameters are effective when the Cache Manager is started or restarted.

Cache Manager Files

The local HFS **/opt/dcelocal/etc/CacheInfo** file that is created during DFS client installation is composed of three fields separated by colons:

- The first field specifies where the DFS global namespace is mounted.
- The second field indicates the minor device number for the local DCE Local File System aggregate where the Cache Manager creates its cache files.

  Note: Although this aggregate is not used with a memory cache, an entry must appear in this field even if memory caching is used for the DFS client.
- The third field specifies the cache size in 1024-byte (1 kilobyte) blocks.

For more information see “Choosing Cache Type, Location, and Size” on page 254.

The Cache Manager creates and maintains the **V***n*, **CacheItems**, and **FileSetItems** files, which are not intended for direct use. (These files exist only on an OS/390 system where disk caching was specified for the Cache Manager. If memory caching is specified, a memory cache is used for all cache information the files contain.)

In OS/390, the **V***n* files (**V** files), the **CacheItems** file, and the **FileSetItems** file reside in a special DCE Local File System aggregate. The aggregate is not mounted for local access or exported for access by DFS clients.

Important Note to Users

In OS/390, you cannot access the **V***n*, **CacheItems**, or the **FilesetItems** through OMVS. The DCE Local File System aggregate file system where the files reside is not locally mounted. The **/opt/dcelocal/var/adm/dfs/cache** directory does not exist. If these files were deleted, **DFSCM** would fail in an indeterminate manner. Always use the commands provided with DFS to alter these files.

Following is a description of the files created by the Cache Manager.

/opt/dcelocal/var/adm/dfs/cache/V*n*
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.

/opt/dcelocal/var/adm/dfs/cache/CacheItems
The Cache Manager uses this binary file to record information about each **V** file, including its file ID number and data version number.

/opt/dcelocal/var/adm/dfs/cache/FilesetItems
This binary file stores the fileset-to-mount point mapping for each fileset accessed. This mapping enables the Cache Manager to respond correctly to commands such as **pwd**.

Cache Manager Features You Can Customize

You can alter the following aspects of the Cache Manager configuration to achieve different levels of performance on different client machines:

Disk or Memory Cache
You can direct the Cache Manager to use machine memory instead of disk space for caching using the **dfsd** command **-memcache** command.

Note: In OS/390, the Cache Manager initialization parameters are the same as the **dfsd** command options described in "dfsd" on page 589.

Cache Location
The DCE Local File System aggregate used for the cache can be changed to take advantage of greater space availability in another aggregate using the second parameter of the **CacheInfo** file described in "Choosing Cache Type, Location, and Size" on page 254.

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. Use the third parameter of the **CacheInfo** file or the **dfsd** command **-blocks** option to alter the size of the **DFSCM** cache.

Chunk Size and Number
You can use several Cache Manager parameters to alter the default size and number of chunks that compose a cache using the **dfsd** command **-chunksize** and **-dcache** options.

With a disk cache, each chunk is called a **V** file. For 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.

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). Refer to the **cm setsetuid** command described in “cm setsetuid” on page 578 for more information on controlling execution of **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. Refer to the **cm setdevok** command described in “cm setdevok” on page 569 for more information on controlling Cache Manager device file processing.

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 by issuing the **cm flush** command described in “cm flush” on page 548.

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. Refer to the **cm lsstores** command described in “cm lsstores” on page 563 and to the **cm resetstores** command described in “cm resetstores” on page 565.

RPC Authentication Levels
The Cache Manager and File Server default authentication levels are such that communications default to the packet integrity RPC, **rpc_c_protect_level_pkt**, 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.

Important Note to Users

In OS/390, you must issue the commands described in this section from OMVS on the OS/390 system where the DFS Cache Manager is being configured. Some of the commands require that you log in as **root**. In OS/390, **root** equates to **UID = 0**. Other commands require no privileges. The necessary privileges are indicated with each command.

Choosing Cache Type, Location, and Size

The Cache Manager default 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** Cache Manager parameter. Refer to “Cache Manager Initialization Parameters” on page 273 for more information on specifying Cache Manager initialization parameters. When the **-memcache** option is used, the Cache Manager does no disk caching, even if the machine has a disk available.

Important Note to Users

During the standard OS/390 DFS installation process the DFS client is installed using a memory cache for installation verification.

The **CacheInfo** file defines the DCE Local File System aggregate 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 installation instructions provide details for creating the **CacheInfo** file.) The **CacheInfo** file contains the following three fields separated by colons.

1. A directory on the local disk where the Cache Manager mounts the DFS global namespace.

   Note: If **/...** is not specified, symbolic links to the global namespace fail.

2. The minor device number for a local DCE Local File System aggregate that serves as the DFS cache for a disk cache. The Cache Manager creates its cache files in this aggregate. There is no default for this entry. The entry must be specified.

   Note: Although this aggregate is not used with a memory cache, an entry must appear in this field even if memory caching is used for the DFS client.

3. 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 minor device number of **999** for the local DCE Local File System aggregate used for the cache files, and a defined cache size of 512 kilobyte blocks (the aggregate must have this many blocks available on its disk(s)):

```
/...:999:512
```

Notes:

1. The **CacheInfo** file contents shown above reflect the contents of the file immediately after DFS is installed.

2. Again, a minor device number must be specified even if memory caching is being used. Refer to "Cache Manager Initialization Parameters" on page 273 for information on specifying **DFSCM** startup parameters by the **_IOE_CM_PARMS** environment variable that includes the **-memcache** option indicating memory caching is active.

3. The **CacheInfo** file is optional. If the **CacheInfo** file does not exist, the following defaults are taken:

   /... - mount directory
   998 - minor device number of the disk cache aggregate
   8000 - number of 1K disk cache blocks

   These defaults may, of course, be overridden by parameters specified in the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file.

Altering Default Parameters for the Cache Manager

The fields in the **CacheInfo** file are the only Cache Manager parameters that you must set. However, you can use the available **dfsd** commands options as Cache Manager initialization parameters to alter several Cache Manager defaults, affecting the way information is cached. Refer to "Cache Manager Initialization Parameters" on page 273 for information on how to specify DFS client initialization parameters. Refer to "dfsd" on page 589 for a description of the available **dfsd** commands options in OS/390.

Following are the configuration parameters that have the greatest effect on cache performance.

Total Cache Size
The amount of disk space or memory available for caching is controlled using the **dfsd** command **-blocks** option described on page 589. Cache size considerations are described later in this section.

Chunk Size
The **dfsd** command **-chunksize** option described on page 591 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
The **dfsd** command **-files** option described on page 590 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 (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 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. The **dfsd** command **-dcache** option described on page 591 controls the number of dcache entries in memory.

DCE single sign-on
The **_EUV_AUTOLOG** environment variable for the **DFSCM** address space determines whether to automatically dce_login a user, from information in the RACF DCE segment, if the user has not been dce_logined yet. Refer to “DFS Client ENVAR File” on page 274 and Appendix A, “Environment Variables in DFS” on page 775 for more information about **_EUV_AUTOLOG** and other **DFSCM** environment variables.

Translation
The **dfsd** command **-translation** option described on page 592 determines whether the DFS client should translate file data between ASCII and EBCDIC code pages.

Note: **dfsbind** and **dfsd** command parameters are specified as Cache Manager initialization parameters on the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file. Parameters specified here take effect the next time the DFS client is restarted.

Disk Cache Configuration

In OS/390, a local DCE Local File System aggregate contains DFS Client (**DFSCM**) disk cache files described in "Cache Manager Files" on page 252.

Note: Although a local DCE Local File System aggregate is used for **DFSCM** cache files, the aggregate is not exported nor are the files created in the aggregate accessible from OMVS.

The second parameter in the **CacheInfo** file identifies the minor device number of the local DCE Local File System aggregate. The **/opt/dcelocal/var/dfs/devtab** entry **define_lfs n** defines a minor device number for the aggregate as described in "Initializing a DCE Local File System Aggregate" on page 167.

DFSCM disk cache resides in a local DCE Local File System aggregate that needs to be allocated and defined on the local OS/390 system. But, **DFSCM** disk caching does not require any DFS server to be running on the OS/390 system where **DFSCM** is running. **DFSCM** and DFS servers are distinct in this regard.

To set up a disk cache for **DFSCM**:

- Allocate and define a local DCE Local File System aggregate.

  Only do the following steps as described in "Initializing a DCE Local File System Aggregate" on page 167:

  1. Modify the **devtab** file

     Add a **define_lfs 999** line (assuming that the DFSCM Disk Cache aggregate has a minor device number of **999** specified in the second field of the **CacheInfo** file). Then add one or more VSAM Linear Data Set names that will make up the DFSCM Disk Cache.
  2. DO NOT modify the **dfstab** file. The DFSCM Disk Cache aggregate will not be exported.
  3. Use JCL to allocate one or more VSAM Linear Data Sets that will make up the DFSCM Disk Cache Logical Volume.

     Use the same data set names that you specified in Step 1.
  4. Run the **newaggr** JCL to initialize the DFSCM Disk Cache as a DCE Local File System aggregate.

     Specify the first parameter of **newaggr** as minor device **999** as follows:

     ```
     // PARM=('//dev/lfs999 8192 1024 -verbose')
     ```

  Note: Do not export the local DCE Local File System aggregate used for **DFSCM** disk caching.

- Update the **DFSCM** initialization parameter **_IOE_CM_PARMS** specification in the **/opt/dfslocal/home/dfscm/envar** file to specify **-dcache**. (Note: **-dcache** is the default for **DFSCM** processing if **-memcache** is not specified.)
- Stop and restart **DFSCM**.

The number of kilobyte blocks available for disk caching out of the aggregate is defined in the third field of the **CacheInfo** file.

The following options are specified in the **_IOE_CM_PARMS** environment variable in the **/opt/dfslocal/home/dfscm/envar** 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:

```
_IOE_CM_PARMS=-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. The following example sets the number of chunks to 2000:

```
_IOE_CM_PARMS=-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}):

```
_IOE_CM_PARMS=-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:

```
_IOE_CM_PARMS=-dcache 250
```

When altering a disk cache configuration, any combination of **dfsd** options is allowed. However, the cache size defined in the **CacheInfo** file or with the **-blocks** option cannot be exceeded with the **-files** or **-chunksize** option.

Memory Cache Configuration

The default chunk size for a memory cache is 8 kilobytes (2^{13}). There is no predefined default for the number of chunks in a memory cache, and, as mentioned previously, the number of dcache entries equals the number of chunks.

If the **-blocks** option is used alone, it overrides the default cache size in the **CacheInfo** file. The Cache Manager divides this value by the default chunk size of 8 kilobytes to calculate the number of chunks and dcache entries. The following example sets the cache size to 5 megabytes (5120 kilobytes); as a result, the number of chunks is set to 640 (5120 divided by 8):

```
_IOE_CM_PARMS=-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:

```
_IOE_CM_PARMS=-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:

```
_IOE_CM_PARMS=-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 causes the Cache Manager to multiply 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 causes the Cache Manager to set the specified values and to multiply 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 sets the number of chunks for a disk cache and is therefore ignored for a memory cache.
- The **-blocks** and **-dcache** options together may cause the Cache Manager to choose one of the values to ignore, or the command may fail.

(See “dfsd” on page 589 for complete information about these options and the **dfsd** command usage.)

Changing Cache Location

The default directory for the Cache Manager's cache is **/opt/dcelocal/var/dfs/adm/cache**. You can change this to a directory on another local DCE Local File System aggregate if more space is available.

You can change the location of the cache by editing the **CacheInfo** file or by using the **-cachedir** option as a Cache Manager initialization parameter. (See “dfsd” on page 589 for more information about the **dfsd** command parameters that are used as Cache Manager initialization parameters.) See “Cache Manager Initialization Parameters” on page 273 for information on how to specify Cache Manager initialization parameters.)

To change the cache location, do the following:

1. Log in as **root** (a user with **UID = 0**) on the system where the DFS client is running.
2. Define a new DCE Local File System aggregate must be on the local disk of the machine. (Refer to “Disk Cache Configuration” on page 257 for **DFSCM** disk cache information.)
3. Use a text editor to change the second field of the **CacheInfo** file (the information between the two colons) to the minor device number used to identify the new DCE Local File System aggregate. The following example shows the **CacheInfo** file identifying a minor device number of 998 for the new disk cache:

   ```
   /...:998:15000
   ```
4. Stop and restart **DFSCM** from the system operator console.
   - First, stop the DFS client if it is running, using the system operator console command:

     ```
     stop dfscm
     ```

     Refer to the *OS/390 MVS System Commands* book, GC28-1781, for more information on the operator console commands.
   - Then, restart **DFSCM** by replying to the system operator console message:

     ```
     *nn BPXF014D FILESYSTYPE DFSC TERMINATED. REPLY 'R' WHEN READY TO RESTART.
     ```

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.

Determining Cache Size

The amount of disk space or memory used for caching depends on several factors. The size of the DCE Local File System aggregate that houses the cache or the amount of memory available on the machine places a limit on cache size.

Note: You should not use more than 75% of the DCE Local File System aggregate for disk cache.

In OS/390, since the DFS client usually serves multiple users, a cache of 60 to 70 megabytes or even larger may be appropriate.

A cache smaller than 5 megabytes can hamper Cache Manager performance as described previously; 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 restarting **DFSCM** address space (you must be logged in as **root** or have write permission to the file to edit it). The **-blocks** option can also be specified as an **_IOE_CM_PARMS** value in the **/opt/dfslocal/home/dfscm/envar** file to override the **CacheInfo** value when **DFSCM** is restarted. The **dfsd** command options that can be specified in the **/opt/dfslocal/home/dfscm/envar** file that take effect when **DFSCM** is restarted are described in the “dfsd” on page 589.

To alter disk cache size without restarting **DFSCM**, use the **cm setcachesize** command. The value remains in effect until **DFSCM** is next restarted 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 restart **DFSCM** to reset the cache size for a memory-caching machine. Refer to “Starting DFS Cache Manager (DFSCM)” on page 126 for information on restarting **DFSCM**.

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 listing 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

/...:998:25000
```

Displaying the Current Cache Size and the Amount in Use

Issue the **cm getcachesize** command on a client machine to view the current size of the cache and the amount in use. On machines that use disk caching, the current cache size may disagree with the default size specified in the **CacheInfo** file if the cache size was changed with the **cm setcachesize** command. Regardless of the type of caching in use, the current cache size may also disagree with the **CacheInfo** file if the size was changed with the **-blocks** option of the **_IOE_CM_PARMS** environment variable specified in **/opt/dfslocal/home/dfscm/envar** file.

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
```

```
DFS is using 13709 of the available 25000 1K byte disk cache blocks.
```

Changing the Disk Cache Size Temporarily

You can reset the disk cache size without starting **DFSCM**. The value remains in effect until you next restart **DFSCM**. To change the disk cache size temporarily, do the following:

1. Log in as **root** on the system where **DFSCM** is running.
2. From OMVS, 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 disk cache size to the amount specified in the **CacheInfo** file.

Resetting the Disk Cache Size to the Default

To reset the disk cache size to the default, do the following:

1. Log in as **root** on the system where **DFSCM** is running.
2. From OMVS, use the **-reset** option with the **cm setcachesize** command to reset the disk cache size to the value specified at the last restart of **DFSCM**, which is either the value in the **CacheInfo** file or the value set with the **-blocks** option of the **dfsd** command specified in **/opt/dfslocal/home/dfscm/envar** file.

   ```
   # cm setcachesize -reset
   ```

   The **-reset** option resets the disk cache size to the value at the last time **DFSCM** was started.

Changing the Cache Size Permanently

To change the cache size permanently, do the following:

1. Log in as **root** on the OS/390 system where **DFSCM** is running.
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:

   ```
   /...:998:25000
   ```

Attention: Be precise when editing the **CacheInfo** file. Use colons to separate the fields in the file; do not include any spaces in the file.

Alternatively, you may add/change the **-blocks** option in the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file.

3. Restart **DFSCM**. Refer to "Starting DFS Cache Manager (DFSCM)" on page 126 for information on restarting **DFSCM**.

Setting File Server Machine Preferences

A replicated DCE Local File System fileset typically has multiple read-only replicas. Each replica provides the same data, but each resides on a different File Server machine. 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 Server machines where that replica resides. If the replica resides on multiple File Server machines, the Cache Manager must choose the machine from which to access the replica.

To choose from among the possible File Server machines, the Cache Manager consults its collection of File Server machine preferences. Each preference consists of the hostname or Internet Protocol (IP) address of a File Server machine and the numerical rank of the machine in the range from 1 to 65,534. The Cache Manager attempts to access the replica from the File Server machine that has the lowest recorded rank. If two machines 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 replica from the machine with the lowest rank (because the machine is currently unavailable, for example), it attempts to access the replica from the machine with the next-lowest rank. It continues in this fashion until it either succeeds in accessing the replica or determines that all machines that house the replica are unavailable.

The Cache Manager stores its File Server preferences in the kernel of the local machine, which causes it to lose 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 machine that it contacts or that houses a replica of a read-only fileset from which it accesses data. The Cache Manager bases its default ranks on IP addresses, as follows:

- If the local machine is also a File Server machine, it receives an initial rank of 5000.
- Each File Server machine in the same subnetwork as the local machine receives an initial rank of 20,000.
- Each File Server machine in the same network as the local machine receives an initial rank of 30,000.
- Each File Server machine 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.

To minimize the chances that multiple machines have the same rank, the Cache Manager adds a random number in the range from 0 (zero) to 15 to each rank that it calculates. For example, while it assigns an initial rank of 20,000 to a File Server machine in the same network as the local machine, the Cache Manager records the actual rank as an integer in the range from 20,000 to 20,015. Because they are based on IP address, default ranks bias the Cache Manager to favor File Server machines that are the same subnetwork or network as the local machine.

To display the current set of File Server machine preferences of the Cache Manager, use the **cm getpreferences** command. By default, the command displays its output on standard output, but you can direct the output to a specified file.

To specify preferences for one or more File Server machines, use the **cm setpreferences** command. Each preference that you specify must consist of the following pair of values:

- The hostname or IP address of the File Server machine whose preference you are setting.
- The rank of the machine in the form of an integer in the range from 1 to 65,534.

As with default ranks that it calculates, the Cache Manager adds a random integer in the range from 0 (zero) to 15 to each rank that you specify.

Use the command options to specify pairs of File Server machines and their respective ranks as follows:

- Use the **-server** option to specify one or more pairs of File Server machines and ranks on the command line. For example, the following command assigns the File Server machines names **fs1.abc.com** and **fs2.abc.com** ranks from 19,000 and 21,000 (plus a random integer from 0 to 15):

  ```
  # cm setp -se fs1.abc.com 19000 fs2.abc.com 21000
  ```

- Use the **-path** option to specify the pathname of the file that specifies pairs of File Server machines and ranks (the **cm getpreferences** command can be used 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 file **/etc/cm.prefs** contains the following lines:

  ```
  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 pairs of File Server machines and ranks from standard input. for example, the following command reads preferences piped to the machine from a user-defined program named **mkprefs**:

  ```
  # mkprefs | cm setp -st
  ```

  The program **mkprefs** generates 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 machine that you specify, the rank you specify replaces the existing rank of the machine.

Each Cache Manager maintains its own collection of preferences, so two Cache Managers can have two different ranks for the same File Server machine.

Cache Managers also maintain preferences for Fileset Location Server machines. Include the **-fldb** option on the **cm getpreferences** and **cm setpreferences** commands to specify that the command applies to Fileset Location Server machines instead of File Server machines.

Displaying File Server Preferences

Issue the **cm getpreferences** command to display Cache Manager preferences for File Server machines:

```
$ cm getpreferences [-path filename] [-numeric]
```

- The **-path** *filename* option specifies a file to which the command is to write its output. Omit this option to display the output on standard output.

- The **-numeric** option directs the command to display the IP address rather than the hostnames of the File Server machines included in the output that it reports. Omit this option to display the hostnames.

The command produces output of the following form for each File Server machine for which the Cache Manager has a recorded preference:

```
hostname          rank
```

In the output, *hostname* is the hostname of a File Server machine, and *rank* is the numeric rank of the machine. Note that *hostname* is replaced with the IP address of the machine if the Cache Manager cannot presently determine the hostname or if the **-numeric** option is included with the command.

Setting File Server Preferences

To set the preferences of the Cache Manager for one or more File Server machines, perform the following steps:

1. Log in as **root** on the machine.
2. Issue the **cm preferences** command to set the preferences of the Cache Manager for one or more File Server machines:

   ```
   # cm setpreferences [-server machine rank ...] [-path filename ] [-stdin]
   ```

   - The **-server** *machine rank* option specifies one or more pairs of File Server machines and their ranks. Separate each machine specification and each rank with one or more spaces.
   - The **-path** *filename* option specifies a file from which the command is to read one or more pairs of File Server machines 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 machines and their ranks from standard input. Separate each machine specification and each rank with one or more spaces.

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 programs allow anyone who uses them 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 **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.

Note that **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

Use the **cm getsetuid** command to determine whether the Cache Manager allows programs from specific filesets to execute with **setuid** permission.

Note: Every program also has a **setgid** bit that, when set, allows a person executing the program to execute with the permissions of the group that owns the program for the duration of its execution. When the **cm setsetuid** command is used, it automatically enables or disables **setgid** permission at the same time. Thus, if **setuid** programs are enabled on a fileset, **setgid** programs are also enabled on that same fileset.

Checking setuid Permission

Issue the **cm getsetuid** command to determine the status of **setuid** programs on specific filesets:

```
$ cm getsetuid [-path {filename | directory_name}...]
```

The **-path** option specifies a file or directory from each fileset whose **setuid** status is to be displayed. Omit this option to display the status for the fileset that contains the current working directory.

The output from this command includes a line for each specified fileset, stating either:

- `no setuid allowed`, indicating that **setuid** (and ***setgid***) programs from the fileset are disabled;
- `setuid allowed`, indicating that **setuid** (and **setgid**) programs from the fileset are enabled; or
- `cm: the fileset on which '`*pathname*`' resides does not exist`, indicating that the pathname specified with the **-path** option is invalid.

Changing setuid Permission

To change **setuid** permission, do the following:

1. Log in as **root** on the OS/390 system where **DFSCM** is running.
2. From OMVS, issue the **cm setsetuid** command to change the status of **setuid** (and **setgid**) programs on specific filesets:

   ```
   # cm setsetuid [-path {filename | directory_name}...][-state {on | off}]
   ```

   - The **-path** option specifies a file or directory from each fileset whose **setuid** status is to be changed. Omit this option to change the status for the fileset that contains the current working directory.
   - The **-state on** option allows **setuid** programs from the indicated filesets to execute with **setuid** permission; the **-state off** option prevents **setuid** programs from the indicated filesets from executing with **setuid** permission. If this option is omitted, **setuid** programs from the specified filesets are allowed to execute with **setuid** permission.

Determining Device File Status

The OS/390 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 **cm setdevok** command sets device file status on a per-fileset and per-Cache Manager basis.

Use the **cm getdevok** command to determine whether the Cache Manager honors device files from specific filesets.

Checking Device File Status

Issue the **cm getdevok** command to determine whether the Cache Manager honors device files on specific filesets:

```
$ cm getdevok [-path {filename | directory_name}...]
```

The **-path** option specifies a file or directory from each fileset about which device file status information is to be displayed. Omit this option to display the status for the fileset containing the current working directory.

The output from this command includes one line for each specified fileset, stating either

- `device files allowed`, indicating that device files from the fileset are honored;
- `device files not allowed`, indicating that device files from the fileset are not honored; or
- `cm: the fileset on which 'pathname' resides does not exist`, indicating that the pathname specified with the **-path** option is invalid.

Changing Device File Status

To change device file status, do the following:

1. Log in as **root** on the system where **DFSCM** is running.
2. From OMVS, issue the **cm setdevok** command to change the status of device files on specific filesets:

   ```
   # cm setdevok [-path {filename | directory_name}...] [-state {on | off}]
   ```

   - The **-path** option specifies a file or directory from each fileset for which device file status is to be changed. Omit this option to change the status for the fileset containing the current working directory.
   - The **-state on** option causes device files from the indicated filesets to be honored; the **-state off** option prevents device files from the indicated filesets from being honored. If this option is omitted, device files from the specified filesets are honored.

Updating Cached Data

When an application program requests new data and the cache is full, the Cache Manager discards some data to make room for the new information. It discards data based on the following two factors.

- If the data is reproducible; information is considered reproducible if it is unchanged from its first retrieval. By definition, data from read-only filesets is always reproducible; data from read/write filesets that was changed by a local application is considered reproducible if the changes are stored to the File Server machine.
- When the application program last referenced the data; data not used for the longest time is discarded first.

Thus, reproducible data not used for the longest time is discarded first. The Cache Manager continues to discard least-recently used (LRU) data in this fashion until there is enough room for the new data.

You can force the Cache Manager to discard, or flush, data cached from files, directories, and filesets. You can flush individual files or directories with the **cm flush** command, or you can flush one or more filesets with the **cm flushfileset** command. Flushing is necessary only in the event of file system problems or for testing purposes. The **cm flush** and **cm flushfileset** commands do not cause the Cache Manager to discard changes to data not written back to the central copies of files. These commands also do not affect data in the buffers of application programs.

The Cache Manager checks once an hour for changes that do not involve tokens, such as the release of a new version of a cached read-only fileset or a name change for any cached fileset. You can force the Cache Manager to notice these changes at other times with the **cm checkfilesets** command, which directs the Cache Manager to revise its table of mappings between fileset names and fileset ID numbers.

Flushing Specific Files or Directories

Issue the **cm flush** command to discard data from specific files or directories:

```
$ cm flush [-path {filename | directory_name}...]
```

The **-path** option names each file or directory that you want to flush from the cache. If a *directory_name* is used, the Cache Manager flushes the name mappings and the blocks associated only with the directory, not with the files in the directory. If this option is omitted, the current working directory is flushed.

Flushing All Data from Specific Filesets

Issue the **cm flushfileset** command to discard data from specific filesets:

```
$ cm flushfileset [-path {filename | directory_name}...]
```

The **-path** option names each file or directory in a fileset whose contents you want to flush. The Cache Manager flushes everything cached from each fileset that contains a specified file or directory. If this option is omitted, the fileset that contains the current working directory is flushed.

Forcing the Cache Manager to Notice Other Fileset Changes

Issue the **cm checkfilesets** command to make the Cache Manager check for changes to information about filesets that contain cached data:

```
$ cm checkfilesets
```

Listing and 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 system operator console when 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 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.

Attention: Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

You can use the **cm statservers** command to determine which File Server machines are failing to respond to the Cache Manager. (See “Checking File Server Machine Status” on page 269 for information about the **cm statservers** command.)

Listing Unstored Data

Issue the **cm lsstores** command to list filesets that contain unstored data that the Cache Manager cannot write back to a File Server machine:

```
$ cm lsstores
```

Discarding Unstored Data

To discard unstored data, do the following:

1. Log in as **root** on the system where **DFSCM** is running.
2. From OMVS, issue the **cm resetstores** command to cancel any further attempts by the Cache Manager to contact unavailable File Server machines. The Cache Manager discards all data that it has been unable to store to such File Server machines.

   ```
   $ cm resetstores
   ```

Checking File Server Machine Status

If the Cache Manager cannot access files or save files that it currently has cached, you can use the **cm statservers** command to determine if one or more File Server machines are currently inaccessible. The command checks the status of each File Server machine with which the Cache Manager has been in contact. The command does not report the reason that a File Server machine is unavailable, but it does list the names of unresponsive machines.

The Cache Manager classifies as unresponsive any File Server machine that meets the following pair of conditions:

- The Cache Manager has been in contact with the File Exporter running on the machine and needs to contact it in the future (for example, the Cache Manager is holding tokens to data on the machine).
- The File Exporter on the machine is not responding to the Cache Manager's periodic probes (implying that it also is not responding to requests for data).

The Cache Manager does not probe all File Server machines in the local cell; it probes only those File Server machines that house data it has cached. Similarly, the **cm statservers** command is concerned only with File Server machines that do not respond to the Cache Manager's probes. You can use the command to check the statuses of File Server machines that meet the first of the previous two conditions and reside in the local cell, in a specific foreign cell, or in any cell.

To check the status of each File Server machine with which the Cache Manager has been in contact, issue the **cm statservers** command:

```
$ cm statservers [{-cell cellname | -all}]
[-fast]
```

The **-cell** *cellname* option specifies the name of a foreign cell whose File Server machines the Cache Manager is to check. The Cache Manager determines the status of each File Server machine with which it has been in contact from the specified cell. Use the **-cell** option or use the **-all** option to direct the Cache Manager to check File Server machines in all cells; omit both options to check the status of each File Server machine the Cache Manager has contacted from the local cell only.

The **-all** option directs the command to check the status of each File Server machine with which it has been in contact, regardless of the cell in which a machine resides. Use the **-all** option or use the **-cell** option to specify a specific foreign cell whose File Server machines the Cache Manager is to check; omit both options to check the status of each File Server machine the Cache Manager has contacted from the local cell only.

The **-fast** option directs the command to display the results of its most-recent probes. The Cache Manager does not probe File Server machines to determine their statuses at the instant the command is issued.

If all of the File Server machines that it probes respond, the Cache Manager displays the following output in response to the command:

```
All servers are running.
```

If one or more of the File Server machines that it probes do not respond, the Cache Manager displays the following output:

```
These servers are still down: hostname
```

where *hostname* is the name of each File Server machine that fails to respond. In a multihomed server environment, the *hostname* corresponds to the machine name that the Cache Manager is currently using to access each File Server machine. The output does not contain multiple machine host names for the same File Server machine.

RPC Authentication Level Configuration

You can set the RPC authentication level for communications between the Cache Manager and File Servers. By default, such communications use the packet level for local cell and packet integrity level for non-local cell access. 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.

Cdmf Privacy
Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value using a lower level of encryption than Packet Privacy. (CDMF is Common Data Masking Facility)

Note: Higher authentication levels do incur some overhead and therefore cause some degradation in performance. Lower security levels, while more efficient, do carry an additional risk of attack.

You can set separate authentication levels for dealing with File Servers in the local cell and for dealing with File Servers in foreign cells. These authentication levels are set through two pairs of values. Each pair of values consists of the following:

- An initial RPC authentication level. This value sets the initial RPC authentication level used by the Cache Manager when it attempts to establish communications with a File Server. The initial level is used as a starting point in negotiating an RPC authentication level with the File Server.
- A minimum RPC authentication level. This value defines a lower bound RPC authentication level for the Cache Manager. Should a File Server request an authentication level below this level, the Cache Manager refuses communications with that File Server.

For a complete description of how the Cache Manager negotiates the RPC authentication level, see “Data Communication Security in DFS” on page 68. In short, each File Server (File Exporter) maintains its own pairs of security values. These pairs set maximum and minimum bounds that control RPC authentication for communications with Cache Managers. As with the Cache Manager, one pair controls communications with Cache Managers within the local cell while the other controls communications with Cache Managers in foreign cells. By default, the RPC authentication settings at the File Server and Cache Manager negotiates to the packet integrity authentication level.

The Cache Manager begins the negotiation by sending an RPC to the File Server at the authentication level determined by its initial RPC setting. The File Manager then replies in one of the following three ways:

- The File Server accepts the RPC authentication level it received (the level fell within the range defined by its upper and lower bounds) and begins the process of sending and receiving fileset data with the Cache Manager.
- The File Server finds that the RPC authentication level is above its upper bound and sends a response to the Cache Manager instructing it to lower its authentication level. If the Cache Manager is currently using an authentication level equal to the Cache Manager's lower bound, the Cache Manager ceases attempts to communicate with the File Server.
- The File Server finds that the RPC authentication level is below its lower bound and sends a response to the Cache Manager instructing it to raise its authentication level.

The following sections detail how to initially configure the Cache Manager RPC authentication levels and how to adjust those levels for a running Cache Manager.

Configuring RPC Authentication Levels: The default Cache Manager and File Server authentication settings are such that they 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_c_protect_level_default** or **default** or **0**: Use the DCE default authentication level.
- **rpc_c_protect_level_none** or **none** or **1**: Perform no authentication.
- **rpc_c_protect_level_connect** or **connect** or **2**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_c_protect_level_call** or **call** or **3**: Authenticate only at the beginning of each RPC received.
- **rpc_c_protect_level_pkt** or **pkt** or **4**: Ensure that all data received is from the expected host.
- **rpc_c_protect_level_pkt_integrity** or **pkt_integrity** or **5**: Authenticate and verify that none of the data transferred has been modified.
- **rpc_c_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.
- **rpc_c_protect_level_cdmf_priv** or **cdmf_priv** or **7**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value using a lower level of encryption than **rpc_c_protect_level_pkt_privacy**.

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.

```
_IOE_CM_PARMS=-initiallocalprotectlevel rpc_c_protect_level_connect -minlocalprotectlevel none \
            -initialremoteprotectlevel 6 -initialremoteprotectlevel pkt_privacy
```

When configuring the authentication levels, any combination of **dfsd** options is allowed.

Changing the RPC Authentication Levels Temporarily: You can reset any of the RPC authentication levels without restarting the Cache Manager by using the **cm setprotectlevels** command. The values you alter remain in effect until you restart the Cache Manager.

To change the authentication levels temporarily:

1. Log in as **root** on the machine.
2. Issue the **cm setprotectlevels** command with the appropriate options.

```
$ cm setprotectlevels [-initiallocalprotectlevel level ]
                      [-minlocalprotectlevel level]
                      [-initialremoteprotectlevel level]
                      [-minremoteprotectlevel level]
```

The various options are defined as follows:

-initiallocalprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell.

-minlocalprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell.

-initialremoteprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells.

-minremoteprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells.

where *level* is the appropriate RPC authentication level. The various levels can be set by specifying a complete string, an abbreviated string, or a corresponding integer value.

The example command does the following:

- Sets the initial authentication level for communications with File Servers in the local cell to packet integrity.
- Sets the minimum authentication level for communications with File Servers in the local cell to packet.
- Sets the initial authentication level for communications with File Servers in foreign cells to packet privacy.
- Sets the minimum authentication level for communications with File Servers in foreign cells to packet privacy.

```
$ cm setprotectlevels -initiallocalprotectlevel pkt_integ -minlocalprotectlevel 4 \
                      -initialremoteprotectlevel 6 -minremoteprotectlevel 6
```

Checking RPC Authentication Levels: You can check the Cache Manager's current RPC authentication levels by using the **cm getprotectlevels** command:

```
$ cm getprotectlevels

Initial protection level in the local cell: value
Minimum protection level in the local cell: value
Initial protection level in non-local cells: value
Minimum protection level in non-local cells: value
```

The various possible output strings for *value* are as follows:

- **rpc_c_protect_level_default** - default
- **rpc_c_protect_level_none** - none
- **rpc_c_protect_level_connect** - connect
- **rpc_c_protect_level_call** - call
- **rpc_c_protect_level_pkt** - packet
- **rpc_c_protect_level_pkt_integ** - packet integrity
- **rpc_c_protect_level_pkt_privacy** - packet privacy
- **rpc_c_protect_level_cdmf_priv** - cdmf privacy.

Controlling the DFS Client Usage of the OS/390 DCE Single Sign-on Function

When the DFS client is invoked (as a result of a file request for a file in the DFS namespace), the current DCE credentials are used to make the request. If there are currently no DCE credentials (or they have expired) when the DFS client is invoked, then the DFS client does one of the following:

- Attempts to use the DCE single sign-on function, or
- Sends the file request as an unauthenticated request.

The IBM supplied **DFSCM envar** file has the environment variable definition **_EUV_AUTOLOG=NO**. This suppresses DCE single sign-on processing for **DFSCM**. To activate the DFS client usage of the DCE single sign-on processing for **DFSCM**, set the environment variable definition to **_EUV_AUTOLOG=** or delete the **_EUV_AUTOLOG=NO** definition from the file. For more information, see “DCE Single Sign-On from the DFS client” on page 30.

Cache Manager Initialization Parameters

In OS/390, DFS Cache Manager initialization parameters can be specified by using one or both of the following methods:

- Local HFS **/opt/dcelocal/etc/CacheInfo** file
- **_IOE_CM_PARMS** environment variable in the **/opt/dfslocal/home/dfscm/envar** file.

Parameters specified in the local HFS **/opt/dcelocal/etc/CacheInfo** file or in the **/opt/dfslocal/home/dfscm/envar** file are effective when the Cache Manager is started or restarted. Refer to “Choosing Cache Type, Location, and Size” on page 254 for information about the **/opt/dcelocal/etc/CacheInfo** file and refer to “DFS Client ENVAR File” on page 274 for information about the **/opt/dfslocal/home/dfscm/envar** file.

Note: The **dfsbind** command options in “dfsbind” on page 586 and the **dfsd** command options in “dfsd” on page 589 are Cache Manager initialization parameters specified by the **_IOE_CM_PARMS**

environment variable. In the OS/390 documentation, any reference to these command options is a reference to a Cache Manager initialization parameter.

DFS Client ENVAR File

When initialized by OMVS, **DFSCM** reads the **envar** file specified as the **_EUV_HOME** variable value in the OS/390 UNIX parameter library **BPXPRM*xx* FILESYSTYPE TYPE(DFSC)** entry. A **_EUV_HOME** variable value must be specified in the BPXPRM*xx* entry for **DFSCM**. The use of the file **/opt/dfslocal/home/dfscm/envar** is recommended for **DFSCM** processing. Refer to “BPXPRMxx Entry for DFS Client” for more information.

The DCE and DFS environment variables that are key to the DFS client as follows. Refer to Appendix A, “Environment Variables in DFS” on page 775 for information on other DCE and DFS environment variables.

```
_EUV_AUTOLOG=NO
_EUV_SVC_MSG_LOGGING=CONSOLE_LOGGING
_IOE_CM_PARMS=-mountfilesystem IOE_DFS_CLIENT_DATA -translation text -memcache
TZ=EST5EDT
NLSPATH=/usr/lib/nls/msg/%L/%N
LANG=En_US.IBM-1047
```

Notes:

1. This reflects the default content of the **/opt/dfslocal/home/dfscm/envar** file after the **dfs_cpfiles** post-installation program is executed.
2. The **_IOE_CM_PARMS** environment variable is where **dfsd** command options are specified in OS/390.

HFS File System Identifying DFSCM

DFS is installed with **_IOE_CM_PARMS=-mountfilesystem IOE_DFS_CLIENT_DATA**. **IOE_DFS_CLIENT_DATA** is the name of the HFS file system used to uniquely identify the DFS client as a physical file system and is the file system mounted at **/...** during **DFSCM** initialization. The HFS file system does not need to exist. The name of the file system can be changed by altering the file system name specified for the **-mountfilesystem** parameter.

BPXPRMxx Entry for DFS Client

DFSCM requires a FILESYSTYPE TYPE(DFSC) entry in the OS/390 UNIX parameter library (parmlib) member BPXPRM*xx* for the system running the DFS client. Refer to the *OS/390 UNIX System Services File System Interface Reference*, SC28-1909, for information on defining a physical file system to OS/390 UNIX using the parmlib member BPXPRM*xx* FILESYSTYPE statement.

To cause OMVS to start **DFSCM**, make the following addition to the parameter member BPXPRM*xx* in the data set that is allocated to the IEFPARM DD statement in the OMVS Started Task PROC:

```
 FILESYSTYPE TYPE(DFSC)
            ENTRYPOINT(IOECMINI)
            PARM('ENVAR("_EUV_HOME=/opt/dfslocal/home/dfscm") /
            >DD:IOEDFSD 2>&1')
            ASNAME(DFSCM)
```

TYPE(DFSC) must be specified as shown or the DFS client fails in an indeterminate manner. **DFSC** is the well known name that is used to identify **DFSCM** as a physical file system and is used in requests issued from **DFSCM** to OS/390 UNIX.

ENTRYPOINT(IOECMINI) must be specified as shown or OMVS will not successfully initialize the DFS client during OMVS initialization.

_EUV_HOME environment variable must be specified in the **FILESYSTYPE** statement. A directory other than **/opt/dfslocal/home/dfscm** may be specified but the use of this directory for the DFS client is recommended.

Note: A trailing slash '/' after the **_EUV_HOME** specification is required. **_EUV_HOME** is a DCE runtime option (not a **DFSCM** parameter). A slash identifies the end of the runtime options and the start of the parameters. If the slash '/' is omitted, it indicates that there are no runtime options and **_EUV_HOME** is not defined because it is treated as a **DFSCM** parameter.

>DD:IOEDFSD 2>&1 must be specified as shown. It assigns standard out (**STDOUT**) and standard error (**STDERR**) for **DFSCM** DCE login processing to the ddname IOEDFSD. The ddname IOEDFSD is required to be in started procedure JCL for the DFS client (refer to “ASNAME Parameter”).

ASNAME Parameter

ASNAME(DFSCM) must be specified as shown. It is the name (**DFSCM**) of the started procedure for the DFS client. The following is an example of **DFSCM** procedure JCL:

```
//DFSCM    PROC REGSIZE=0M,            REGION SIZE
//         OUTCLASS='X'                HELD OUTPUT CLASS
//GO       EXEC PGM=BPXVCLNY,REGION=&REGSIZE,TIME=1440
//SYSPRINT DD     SYSOUT=&OUTCLASS
//IOEDFSD  DD     SYSOUT=&OUTCLASS         DFSCM    SYSOUT
//IOELOGIN DD     SYSOUT=&OUTCLASS         DFSCM    SYSOUT
//CEEDUMP  DD     SYSOUT=&OUTCLASS
//DFSDDUMP DD     SYSOUT=&OUTCLASS
//LOGNDUMP DD     SYSOUT=&OUTCLASS
//SYSUDUMP DD     SYSOUT=&OUTCLASS
//*
```

Notes:

1. **DFSCM** parameters can be supplied in the **CacheInfo** file as described in “Choosing Cache Type, Location, and Size” on page 254 or in the **envar** file described in “DFS Client ENVAR File” on page 274.
2. The **_IOE_CM_PARMS** environment variable is where **dfsd** command options are specified in OS/390.

Mount Point for DFS Global Namespace

The DFS global namespace must be mounted on a local disk directory. In OS/390, the local directory where the global namespace is mounted must be **/...**. (The **/...** is usually created during the DFS installation.) The global namespace is identified by the local file system name specified as the **-mountfilesystem** parameter in the **_IOE_CM_PARMS** environment variable specified in the **/opt/dfslocal/home/dfscm/envar** file. During DFS Client (**DFSCM**) initialization, this local file system name is mounted on the local directory identified by the first field of the **CacheInfo** file.

Chapter 14. Configuring the Backup System

Important Note to Users

There are exceptions to the information in this chapter for OS/390 DFS. It is important to note the following while reading this chapter:

- On OS/390 DFS, the following information applies to DCE Local File System filesets only. For information on backing up your OS/390 HFS file system, see the *OS/390 UNIX System Services User's Guide*, SC28-1891.
- The following **bak** commands are not applicable against OS/390 tapes used by the OS/390 backup tape coordinator:
  - **bak labeltape** command
  - **bak readlabel** command
  - **bak scantape** command.
- The **fms** command is not available.
- For OS/390 DFS, the Tape Coordinator is run in the background as part of DFS address space. Messages are printed on STDOUT and prompts for tape mounts and dismounts go to the OS/390 operator's console.
- The **TapeConfig** file described in this chapter is not required by the Backup Tape Coordinator.

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 re-creates the filesets as they were before a specific date. Because both DCE Local File System filesets and exported non-Local File System file systems have entries in the Fileset Location Database (FLDB), the DFS Backup System can be used with both types of filesets.

This section introduces the DFS Backup System. It describes configuration issues related to the performance of backup and restore operations. Chapter 15, “Backing Up and Restoring Data” provides specific details about listing information from the Backup Database, backing up and restoring data, and administering the Backup Database. Refer to this section to configure the Backup System and to prepare it for backing up and restoring data; refer to Chapter 15, “Backing Up and Restoring Data” to back up and restore data.

Introduction to the Backup System

With the DFS Backup System, you control many aspects of the backup process, including how often backups are performed, which filesets are backed up, and whether full or incremental backups are made. A dump or dump set is the result of performing a backup operation; it includes data from all 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 have 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 the most-recent full dump of the fileset. It then restores the changes to the fileset from any incremental dumps that were made since the last full dump. Two types of restores are possible:

- A full restore, which re-creates the fileset as it was at its last dump, including changes from both the last full dump and any incremental dumps that were made since the full dump; and
- A date-specific restore, which re-creates the fileset as it was at the time of its last dump before an indicated date (the Backup System restores the changes from any incremental dumps performed before the specified date, so the fileset is current according to that date).

The Backup System can be used to dump and restore data between different types of file systems. For example, data dumped from a DCE Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset. Likewise, data dumped from a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System 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 Local File System fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Important Note to Users

The **fts dump** and **fts restore** commands are not available for OS/390 HFS filesets or RFS filesets in OS/390 DFS.

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 4, “DFS Configuration Issues” 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.

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,
- Must be configured as a Tape Coordinator machine, and
- Must run one instance of the **butc** (BackUp Tape Coordinator or just Tape Coordinator) process for each tape drive.

The ***butc*** program must be active when you issue a **bak** command that involves either a tape drive or an operation being performed by a tape drive. For optimum efficiency, run several tape drives (and their Tape Coordinators). Start one **butc** process on a Tape Coordinator machine for each tape drive attached to the machine. Each Tape Coordinator controls the behavior of its associated drive and accepts service requests from the ***bak*** program.

A Tape Coordinator ID (TCID) identifies a Tape Coordinator; each TCID must be unique in the Backup System of the local cell. When you issue **bak** commands, specify a Tape Coordinator by specifying its TCID with the **-tcid** option. Depending on the command, the ***bak*** or ***butc*** program contacts one or more of the following — the Backup Database (by way of the Backup Server), the FLDB, or Fileset Server processes.

Fileset Families and Fileset Family Entries

When creating backups, you copy groups of filesets, known as *fileset families*, to tape. A fileset family includes all of the filesets that you want to dump together onto the same tape (or tapes, if the fileset family contains a large number of filesets). All of the filesets in a fileset family are dumped to tape with the same frequency (for example, once a day or once a week).

Fileset family entries (also referred to as *fileset entries*) define the filesets included in a fileset family. Each entry has three fields — the File Server machine name, the aggregate name, and the fileset name. Because filesets can be moved from File Server to File Server, the first two fields are usually designated with a **.*** wildcard, so a person backing up the filesets does not need to know the File Server machine and aggregate on which they reside. The last field (fileset name) is often specified with a regular expression pattern that matches certain fileset names.

With the Backup System, regular expression characters and the **.*** wildcard can be used in many arguments. (See “Defining Fileset Families and Fileset Family Entries” on page 288 for a description of these characters and their interpretations.)

Dump Hierarchies and Dump Levels

A dump hierarchy is a logical structure that helps define the relationship between full and incremental dumps. As mentioned previously, an incremental dump includes only the files that changed since the fileset was last dumped. When creating an incremental dump, the Backup System uses a previous dump (known as the dump's parent) to serve as a reference point on which to base the incremental dump.

A dump set is the product of dumping a fileset family at a certain dump level (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 period (*fileset_family_name***.***dump_level*).

Each dump level can be associated with an expiration date that specifies when a tape containing data from that dump level can be overwritten. The expiration date is transferred to any backup tape that contains a dump made at that level. When dumping to tape, the system checks the tape for an expiration date. If the tape's expiration date has not expired (if it is in the future), the system does not overwrite the tape; if no expiration date is defined for the tape or if the tape's expiration date has expired (if it is in the past), the system overwrites the tape (but only with a dump set of the same name).

Command and Monitoring Windows

A single terminal session can be used to issue **bak** commands to the Tape Coordinators on all Tape Coordinator machines. The session corresponds to a command shell called the *command window*, which can be opened and closed without affecting the Tape Coordinators. This session can be run from any machine in the cell. Multiple command windows can be used, but they are not necessary.

A separate terminal session must be used for each Tape Coordinator and associated tape drive running on a machine; a terminal session of this type is referred to as a Tape Coordinator's monitoring window. The window must be a connection to the Tape Coordinator machine whose Tape Coordinator and tape drive it is monitoring. The Tape Coordinator runs in the foreground, so no further commands can be issued in the monitoring window. The monitoring window must remain open while the Tape Coordinator runs so that you can see the Tape Coordinator's prompts.

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 restoring filesets to a File Server machine.
- 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 section 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.

CAUTION:
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 section.

Standard Information in this Section

The following sections present standard options and arguments common to many of the commands described in this section. They also present some common operations repeated throughout this section.

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this section. If an option or argument is not described with a command in the text, a description of it appears here. See Chapter 19, "Distributed File Service Commands" on page 401 for complete details about each command.

- The **-tapehost** *machine* option is the DCE pathname of the machine (for example, **/.../abc.com/hosts/bak1**) for which a Tape Coordinator is to be added.

- 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 **.** (period). 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 (sequentially) 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 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 **.** (period). 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 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:

    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 use the keyword **at** to represent a specific date and, optionally, time to use as the expiration date. Absolute expiration dates are expressed using one of the following formats:

    at *mm*/*dd*/*yy* [*hh*:*mm*] | **at** *mm*/*dd*/*yyyy* [*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.

  Note: Valid values for *yy* are 00 to 37, which are interpreted as the years 2000-2037, and 70 to 99, which are interpreted as 1970-1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038-2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038. For a *yyyy* specification, valid values for *yyyy* are 1970-2069. However, values between 2038-2069 are reduced to 2038.

If specified, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The default time is 00:00 (12:00 a.m.).

If a value is entered for the year (*yy*) that falls between the range of 38 and 69, a random time is generated for the default time.

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 (zero) is used to execute the command by default.

Standard Commands and Operations

The following sections describe commands and operations that are used frequently in this guide. If a command or operation is described in detail here, it generally is not described in depth in later sections where it is used.

Important Note to Users

The **TapeConfig** file described in this section is not required by the Backup Tape Coordinator on OS/390 DFS.

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 **/opt/dcelocal/var/dfs/backup/TapeConfig** file and the Backup Database (using the **bak addhost** command). See “Configuring the Backup System” on page 284 for a description of these operations. 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 WRITE (**w**) and EXECUTE (**x**) permissions on the **/opt/dcelocal/var/dfs/backup** directory, which is the directory in which the Tape Coordinator creates its **TL** (log) and **TE** (error) files.
2. To start the Tape Coordinator on MVS, issue the OS/390 operator **MODIFY** command as follows:

   ```
   modify dfs,start butcnn
   ```

   where *nn* can be 01 through 08.

Stopping a Tape Coordinator: When you are finished using a Tape Coordinator, you should stop it from running. To stop a Tape Coordinator process from the Tape Coordinator's monitoring window, enter an interrupt signal (<**Ctrl-c**> or its equivalent) in the monitoring window.

Important Note to Users

In an OS/390 DFS environment, the notation <**Alt-c**>**c** followed by pressing the Enter key is used to indicate an interrupt signal.

To stop the Tape Coordinator on MVS, enter the OS/390 operator **MODIFY** command as follows:

```
modify dfs,stop butcnn
```

where *nn* can be 01 through 08.

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. On OS/390 DFS, the tape size and end-of-file mark size are not used. On OS/390 DFS, a dump is treated as one volume, even if it physically spans multiple tape volumes. 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/fms** 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.

Important Note to Users

The **fms** command is not available in OS/390 DFS.

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 WRITE (**w**), EXECUTE (**x**), and INSERT (**i**) 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/rmt1h** is 2,136,604,672 bytes; the EOF mark size for the drive is 1,910,220 bytes.

   ```
   $ fms /dev/rmt1h

   wrote block: 130408
   Finished data capacity test - rewinding
   wrote 1109 blocks, 1109 file marks
   Finished file mark test
   Tape capacity is 2136604672 bytes
   File marks are 1910220 bytes
   ```

Using the Interactive Interface: You can use the **bak** commands in regular command mode or in interactive mode.

To enter interactive mode, enter the **bak** command at the shell prompt:

```
$ bak
```

To stop interactive mode, enter the **quit** command at the `bak>` prompt:

```
bak> quit
```

If you use interactive mode, note the following:

- You do not need to enter the word **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 section 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.

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 sections 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 15, “Backing Up and Restoring Data” for information on inspecting the status of these prerequisites. If all of the prerequisites are met, turn to the appropriate section for information on using the Backup System.

Configuring a Tape Coordinator Machine for OS/390

Setting up a Tape Coordinator machine consists of using **bak** commands to configure the Tape Coordinators for the machine.

When you initially configure a Tape Coordinator machine, prepare the tape drives.

Perform the following steps:

1. If you are configuring a client machine as a Tape Coordinator machine, verify that the **/opt/dcelocal/var/dfs** and **/opt/dcelocal/var/dfs/backup** directories exist on the machine. Create the directories if they do not already exist.
2. Verify that the individuals who are to use the Backup System are included in the appropriate administrative lists (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.
3. Verify that the **bakserver** process is running on the cell's Backup Database machines. If necessary, issue the **bos status** command to check.

4. A VSAM file should be created to hold a record from each dump on the MVS system. The record contains the tape name, the dump number and the tape label. The tape label consists partially of the tape's expiration date. A new record is added each time a new label is created and written to a tape. When the tape becomes expired based on its expiration date, the tape name becomes eligible for reuse. The record is deleted from the VSAM data set and the dump information is deleted from the backup data base. This process simulates the process available under DFS on UNIX.

   To create the VSAM data set you can submit the job in the defined installation name *xxx*.**SIOESAMP**, member **BUTCVSAM**. You may wish to change some of the parameters in the job, such as the *volser* and *space* values. Job **BUTCVSMP** in the defined installation name, *xxx*.**SIOESAMP**, will print the VSAM data set.

   BUTC will print an error message if the VSAM data set does not exist but the dump process will continue to process correctly. However, no expired backup records will be deleted from the backup data base.

Steps Required for a Client-Only Machine: You must perform the following steps to configure a DCE client that is not a DFS server machine of some type (for example, a File Server machine or a Backup Database machine) as a Tape Coordinator machine. For a client-only machine, perform these steps before you perform the steps in “Steps Required for All Machines” on page 286; 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 INSERT (**i**) 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 “Steps Required for All Machines.”

Steps Required for All Machines: You must perform the following steps to configure any machine as a Tape Coordinator machine. If the machine to be configured is a DCE client but not a DFS server machine of some type, you must perform all of the steps in “Steps Required for a Client-Only Machine” on page 285 before performing the steps in this section. If the machine is configured as some type of DFS server machine, you do not need to perform the steps in “Steps Required for a Client-Only Machine” on page 285.

Important Note to Users

The **TapeConfig** file described in this section is not required by the Backup Tape Coordinator on OS/390 DFS.

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 WRITE (**w**) and EXECUTE (**x**) permissions on the *dcelocal***/var/dfs/backup** directory (the directory in which you must create the **TapeConfig** file).
3. Create the *dcelocal***/var/dfs/backup/TapeConfig** file on the machine with a text editor. Use a single line in the file for each tape drive attached to the Tape Coordinator machine, recording the following information:

- The tape size of the tapes to be used in the drive. The Tape Coordinator uses this capacity as the size of all tapes used in the drive. It is recommended that you use a number 10 to 15% lower than the actual tape capacity to allow for tape variations. The following abbreviations can be used for the tape size unit of measurement (the default is kilobytes); do not leave a space between the number and the letter.
  - Kilobytes: k or K (for example, 2k or 2K)
  - Megabytes: m or M (for example, 2m or 2M)
  - Gigabytes: g or G (for example, 2g or 2G)
- The EOF mark size for the type of tape to be used in the drive. The Backup System appends an EOF mark after each fileset dumped to tape. The size of this mark can affect the amount of space available for backup data. The EOF mark size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. It is recommended that you increase the actual file mark size by 10 to 15% to allow for tape variations.

  If you do not specify a unit of measurement, the default unit used for the EOF size is bytes (*not* kilobytes, as for tape capacity). To indicate other units, use the same abbreviations as for tape capacity.
- The device name of the tape drive. The format of this name varies with each operating system. For example, in the UNIX operating system, a valid device name is **/dev/rst0**.
- The TCID for the Tape Coordinator associated with the drive. The Backup System can track a maximum of 1024 tape drives; legal values are integers from 0 to 1023. You do not have to assign the numbers in sequence, and you can skip numbers. The TCID for any Tape Coordinator must be unique among all TCIDs in the local cell.

  Because the **bak** commands that require you to specify a TCID always use a default TCID of 0, assign a TCID of 0 to the Tape Coordinator for the drive that you will use most often; this enables you to omit the **-tcid** option as often as possible.

If you do not know the tape size or the EOF mark size for the tape drive, determine them by using the **fms** command, as described in “Determining Tape Size and End-of-File Mark Size” on page 283.

Following is an example the contents of the **TapeConfig** file for a machine with two drives. The tape size for each drive is 2 gigabytes; the EOF mark size for each drive is 1 megabyte. The respective TCIDs of the two drives are 0 and 1.

```
2g 1m /dev/rmth0h 0
2G 1M /dev/rmth1h 1
```

4. Ensure that the **bak** and **butc** binary files are stored on the local machine. The **bak** file should be stored in the *dcelocal*/**bin** directory; the **butc** file should be stored in the *dceshared*/**bin** directory (a symbolic link to the file may exist from the *dcelocal*/**bin** directory).
5. Verify that the individuals who are to use the Backup System are included in the appropriate administrative lists (see “Privileges Required to Use the Backup System” on page 280); if necessary, issue the **bos lsadmin** command to check. You also need to ensure that you are included in the **admin.bak** list to issue the **bak addhost** command that follows. To add someone to a list, issue the **bos addadmin** command.
6. Verify that the **bakserver** process is running on the cell's Backup Database machines. If necessary, issue the **bos status** command to check.
7. ***Optional***: At this point, you can issue the **bak** command at the system prompt to enter interactive mode. The advantages of interactive mode are described in “Using the Interactive Interface” on page 283. 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.

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 period 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 period 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 DCE pathname of the File Server machine (for example, **/.../abc.com/hosts/fs1**) that houses the filesets. You can 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 exact 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 ^ (caret) 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 **.** (period) 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, **"user\..*\.bak"** or **user\\..*\\.bak** to indicate all of the filesets that begin with the prefix **user.** and end with the extension **.bak**. Otherwise, the command shell attempts to resolve the regular expression characters rather than pass them to the **bak** command interpreter for resolution. Note that the **.*** notation is interpreted as a single wildcard that must be surrounded with double quotes in noninteractive mode (**".*"**). Characters specified in regular expressions are case sensitive.

All fileset family names must be unique within the Backup Database of the local cell. Create and delete fileset families with the **bak addftfamily** and **bak rmftfamily** commands. Create and delete fileset family entries with the **bak addftentry** and **bak rmftentry** commands.

Suggestions for Creating Fileset Family Entries: The **bak addftentry** command has arguments that correspond to the three fields in a fileset family entry: **-server** (for the File Server machine name field), **-aggregate** (for the aggregate name field), and **-fileset** (for the fileset name field). By combining these arguments in different ways, you can create fileset entries for different groupings of filesets. Table 10 summarizes some suggested groupings.

Table 10. 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	The machine name with **-server** and the wildcard for **-aggregate** and **-fileset**.	**/.../abc.com/hosts/fs1** **“.*”** **“.*”**
Filesets on aggregates of the same name	The aggregate name with **-server** and the wildcard for **-aggregate** and **-fileset**.	**“.*”** **/dev/lfs1** **“.*”**
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/lfs2** **“.*”**
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/lfs3** **“user\..*”**
Every fileset on one aggregate with a commons 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/lfs4** **“sys.*\.readonly”**

Include in a fileset family only those filesets that you wish to dump to the same tape at the same time (for example, weekly or daily) and in the same manner (fully or incrementally).

The two main types of fileset families are those based on a common fileset location (File Server machine and aggregate) and those based on similar contents (as reflected by a fileset name). Because filesets can be moved between machines and aggregates, use name-based fileset family entries rather than location-based ones. For name-based entries, specify the **.*** wildcard for the **-server** and **-aggregate** arguments of the **bak addftentry** command.

Adding a Fileset Family to the Backup Database: To add a fileset family to the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak addftfamily** command to create the fileset family. The fileset family remains empty until you use the **bak addftentry** command to define entries in it.

   ```
   $ bak addftfamily -family fileset_family_name
   ```

Adding a Fileset Family Entry to a Fileset Family: To add a fileset family entry to a fileset family, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. ***Optional:*** At this point, you can issue the **bak** command at the system prompt to enter interactive mode. The advantages of interactive mode are described in “Using the Interactive Interface” on page 283. The following commands assume regular (not interactive) command mode is used.
3. 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 are the File Server machines that house the filesets to be included in the entry. Legal values for a specific machine are the machine's DCE pathname, the machine's hostname, or the machine's IP address. You can also specify the regular wildcard expression (.*), which matches all machine names.
   - The **-aggregate** *name* option is the device name or aggregate name of the aggregate that houses the fileset to be included in the entry. The **.*** wildcard can be used to match the name of any aggregate.
   - The **-fileset** *name* option is the name of the fileset to be included in the entry. The **.*** wildcard or any of the regular expression characters described previously can be used to match the names of multiple filesets.

Deleting Fileset Families from the Backup Database: To delete a fileset family from the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak rmftfamily** command to delete each fileset family that you no longer need. It is not necessary to delete the fileset entries in each fileset family first; they are deleted automatically when the family is removed.

   ```
   $ bak rmftfamily -family fileset_family_name...
   ```

Deleting a Fileset Family Entry from a Fileset Family: To delete a fileset family entry from a fileset family, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak lsftfamilies** command to determine the index number of the entry that you want to delete. This is necessary only if the fileset family contains multiple entries, in which case this command is used to list the entries; if the family contains a single entry, the index is 1.

   ```
   $ bak lsftfamilies -family fileset_family_name
   ```
3. Use the **bak rmftentry** command to delete the entry:

   ```
   $ bak rmftentry -family fileset_family_name -entry fileset_entry_index
   ```

   The **-entry** *fileset_entry_index* option is the index for the entry that you want to delete.

Defining a Dump Hierarchy of Dump Levels

A dump hierarchy consists of one or more full dump levels and any incremental dump levels that you create with the **bak adddump** command. The dump levels define how fileset families are to be dumped; all fileset family entries in a fileset family are dumped at the same time and in the same way (fully or incrementally). A dump of a fileset family at a particular dump level produces a dump set. To create a dump set, specify the name of the fileset family and the level at which that family is to be dumped when you initiate the dump with the **bak dump** command. (See Chapter 15, “Backing Up and Restoring Data” 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.
- 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 without question. However, if the tape's expiration date has not expired, the system refuses to overwrite the tape. The operator can allow the system to overwrite the tape if they reply **U** to the system request to use the tape or unload it.

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 **.** (period) 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 period. 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 15, "Backing Up and Restoring Data" for more information on the **bak lsdumps** command.)

Examples of Dump Hierarchies: Following are examples of two possible dump hierarchies. Each hierarchy backs up data in a different way.

The first dump hierarchy is used to back up user data (data from user filesets). Because this data changes frequently, it is dumped at the end of each working day, starting with a full dump at the beginning of the week (Sunday) and continuing with incremental dumps Monday through Friday. Each incremental dump refers to the full dump as its parent, rather than to the previous day's incremental dump. As a result, each incremental dump contains all of the data that has changed since the full dump was performed. The following commands are used to establish this dump hierarchy:

```
$ bak adddump /sunday
$ bak adddump /sunday/monday
$ bak adddump /sunday/tuesday
$ bak adddump /sunday/wednesday
$ bak adddump /sunday/thursday
$ bak adddump /sunday/friday
```

You can use the **bak lsdumps** command to display the dump hierarchy. In the following example, `/sunday` is the full dump used for the hierarchy:

```
$ bak lsdumps
```

```
/sunday
      /monday
      /tuesday
      /wednesday
      /thursday
      /friday
```

The second dump hierarchy is used to back up filesets containing system binary files. Because these files do not change often, they are backed up only once a week, starting with a full dump at the beginning of the month and followed by an incremental dump at the beginning of each subsequent week. Each weekly dump refers to the previous week's dump as its parent, rather than to the initial full dump. Therefore, each weekly dump contains only those files that changed in the past week, rather than everything that changed since the full dump was performed. The following commands establish this dump hierarchy:

```
$ bak adddump /month
$ bak adddump /month/week1
$ bak adddump /month/week1/week2
$ bak adddump /month/week1/week2/week3
$ bak adddump /month/week1/week2/week3/week4
```

You can use the **bak lsdumps** command to display the dump hierarchy. The following example shows `/month` as the full dump for the hierarchy:

```
$ bak lsdumps

/month
      /week1
            /week2
                  /week3
                        /week4
```

Defining a Dump Level: To define a dump level, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak adddump** command to define one or more dump levels:

   ```
   $ bak adddump -level dump_level...[-expires date...]
   ```

Changing a Dump Level's Expiration Date: To change a dump level's expiration date, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak setexp** command to set the expiration dates of one or more dump levels:

   ```
   $ bak setexp -level dump_level...-expires date...
   ```

Deleting a Dump Level: To delete a dump level, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak rmdump** command to delete a dump level. All dump levels for which the level serves as the parent, either directly or indirectly, are also deleted automatically.

   ```
   $ bak rmdump -level dump_level
   ```

Labeling Tapes on OS/390

The OS/390 operating system labels tapes for you. OS/390 users do not need to label tapes.

Adding and Removing Tape Coordinators

As mentioned in "Configuring a Tape Coordinator Machine for OS/390" on page 284, 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. Any **bak** commands that involve a tape drive have a **-tcid** option for that purpose.

The Backup System can track a maximum of 1024 tape drives. Valid TCIDs are the integers from 0 (zero) to 1023. You do not have to assign the numbers in sequence, and you can skip numbers. The drive with the TCID of 0 (zero) is used by default.

Enter the **bak addhost** command to add an entry for a Tape Coordinator to the Backup Database. Enter the **bak rmhost** command to remove an entry for a Tape Coordinator from the Backup Database. Use the **bak lshosts** command to list the Tape Coordinators that have entries in the Backup Database.

Adding a Tape Coordinator

To add a Tape Coordinator, do the following:

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. Choose the TCID for the drive. Enter the **bak lshosts** command to check previously assigned TCIDs:

   ```
   $ bak lshosts
   ```

4. Enter the **bak addhost** command to define an entry in the Backup Database for the Tape Coordinator:

   ```
   $ bak addhost -tapehost machine [-tcid tc_number]
   ```

Removing a Tape Coordinator

To remove a Tape Coordinator, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bak rmhost** command to delete the entry for the Tape Coordinator from the Backup Database:

   ```
   $ bak rmhost [-tcid tc_number]
   ```

Chapter 15. Backing Up and Restoring Data

Important Note to Users

There are exceptions to the information in this chapter for OS/390 DFS. It is important to note the following while reading this chapter:

- For OS/390 systems, the following information applies to DCE Local File System filesets only. For information on backing up your OS/390 HFS data sets see the *OS/390 UNIX System Services User's Guide*, SC28-1891.
- The following **bak** commands are not applicable against MVS™ tapes used by the MVS backup Tape Coordinator:
  - **bak labeltape**
  - **bak readlabel**
  - **bak scantape**.
- The **butc** command is not available.
- In OS/390 DFS, the Tape Coordinator is run in the background as part of DFS address space. Messages are printed on STDOUT and prompts for tape mounts and dismounts go to the OS/390 operators console.
- The **TapeConfig** file described in this chapter is not required by the Backup Tape Coordinator.

Once the DFS Backup System is properly configured, it can be used to help automate the process of making backup copies of both DCE Local File System and non-Local File System 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 section 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 section assume that the Tape Coordinator machines are properly configured, the fileset families and fileset entries are defined, the dump hierarchy is defined, and the required tapes are labeled, as necessary. Chapter 14, “Configuring the Backup System” described configuration of the Backup System in detail; make sure the Backup System is properly configured according to the guidelines detailed in that section before attempting to use it to perform any of the operations described in this section.

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 10, “Making Filesets and Aggregates Available”) 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 “bos” on page 475 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. (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:
- The Backup System scans the FLDB to match the wildcards from each fileset entry and generates a complete list of the filesets to be included in the dump. If duplicates are found, they are not dumped; only one occurrence of any fileset is included.
- The Backup System reads the label on the tape in the drive to verify that the tape name is acceptable and that the tape does not contain an unexpired expiration date.
- The system transfers the list of filesets to be backed up to the appropriate Fileset Server processes, which determine which data in the filesets was modified after the date and time of the last dump at the **/sunday** level.
- The designated Tape Coordinator puts the gathered data onto tape; the expiration date and other information associated with the dump are stored in the tape's label, and a unique dump ID number is

assigned to the dump. If one tape is not large enough to hold the entire dump set, the Backup System prompts the operator to place additional tapes in the drive, as needed.

Standard Information in this Section

The following are standard options and arguments common to many of the commands described in this section. Some common operations repeated throughout this section are also presented.

Standard Options and Arguments

The following options and arguments are used with many of the commands described in this section. If an option or argument is not described with a command in the text, a description of it appears here. See Chapter 19, “Distributed File Service Commands” on page 401 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 consist of a name preceded by a single / (slash) (for example, **/full**).
  - Incremental dumps 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 sequentially 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 (zero) is used to execute the command by default.

Standard Commands and Operations

The following commands and operations are used frequently in this section. If a command or operation is described in detail here, it generally is not described in depth where it is used later.

Starting a Tape Coordinator

Important Note to Users

The **TapeConfig** file described in this section is not required by the Backup Tape Coordinator on OS/390 DFS.

Before performing a backup or restore operation, you must install at least one tape drive on a Tape Coordinator machine and define its Tape Coordinator in both the **/opt/dcelocal/var/dfs/backup/TapeConfig** file and the Backup Database, as described in Chapter 14, “Configuring the Backup System” 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 WRITE (**w**) and EXECUTE (**x**) permissions on the **/opt/dcelocal/var/dfs/backup** directory (the directory in which the Tape Coordinator creates its **TL** (log) and **TE** (error) files).

2. For OS/390 DFS, the Tape Coordinator is run in the background in the DFS address space. To start the Tape Coordinator, use the OS/390 operator **MODIFY** command:

   ```
   modify dfs,start butcnn
   ```

   where nn can be 01 through 08.

Stopping a Tape Coordinator: When you are finished using a Tape Coordinator, you should stop it from running. To stop a Tape Coordinator process from the Tape Coordinator's monitor window, enter an interrupt signal (<**Ctrl-c**> or its equivalent) in the monitoring window.

To stop the Tape Coordinator on OS/390 DFS, use the OS/390 operator **MODIFY** command:

```
modify dfs,stop butcnn
```

where nn can be 01 through 08.

Using the Interactive Interface: You can use the **bak** commands in regular command mode or in interactive mode.

To enter interactive mode, enter the **bak** command at the shell prompt:

```
$ bak
```

To leave interactive mode, enter the **quit** command at the `bak>` prompt:

```
bak> quit
```

If you use interactive mode, the following guidelines apply.

- 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 section 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.

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**.

Important Note to Users

When OS/390 DFS uses MVS/ESA™ services to control and dynamically allocate tape drives, the **bak** commands: **bak labeltape**, **bak readlabel**, and **bak scantape** are not necessary in OS/390 DFS and are not available.

See “bak” on page 408 for more information on the **bak** commands.

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 following sections and Chapter 14, “Configuring the Backup System” for more information on these commands.)

Checking the Status of a Background Operation: You can check the status of a command that is executing in the background by:

- Looking in the monitoring window for output from the command,
- Entering the **bak status** command, or
- 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)` 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` for a restore operation initiated with the **bak restoreft** or **bak restoredisk** command.
  - `Labeltape (tape_label)` for a tape labeling operation started with the **bak labeltape** command (*tape_label* is the label being placed on the tape).
  - `Scantape` for a tape scanning operation initiated with the **bak scantape** command.
  - `SaveDb` for a database saving operation initiated with the **bak savedb** command.
  - `RestoreDb` 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.
- One of the following messages reporting additional status information about the operation. No message is displayed if the operation is proceeding normally.
  - The `[abort request]` message is displayed if the **kill** command was issued but the operation is not yet cancelled.

- The [abort sent] message is displayed if the operation is cancelled but its execution is not yet stopped.
- The [operator wait] message is displayed if the Tape Coordinator is waiting for the operator to respond to an operator console reply prompt.
- The [drive wait] message is displayed if the Tape Coordinator is waiting for the operator to insert a tape in the drive or waiting for a drive to be free.

The following example shows the status of the operation currently being performed by the Tape Coordinator whose TCID is 0:

```
$ bak status
```

```
Dump (usersys.monday): 312105 Kbytes transferred, fileset user.terry.
```

(See “Displaying Operations in Interactive Mode” on page 315 for information about the **bak jobs** command.)

Listing Backup Information

The following commands can be used to list information about the Backup Database:

bak verifydb	Checks the status of the Backup Database.
bak lsftfamilies	Lists fileset families and fileset family entries.
bak lsdumps	Lists the entries in the dump hierarchy.
bak dumpinfo	Displays information about specific backups.
bak lshosts	Lists Tape Coordinator IDs.
bak ftinfo	Displays the dump history for a fileset.

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 can be used.
```

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 parameter to view all of the fileset families and entries defined in the Backup Database.

Following is an example of the output from this command:

```
$ bak lsftfamilies usersys
```

```
Fileset family usersys:
 Entry 1: server .*, aggregate .*, filesets: user.*
 Entry 2: server .*, aggregate .*, filesets: sys.*
```

Listing Entries in the Dump Hierarchy

Issue the **bak lsdumps** command to view the entries in the dump hierarchy.

```
$ bak lsdumps
```

See page 438 for an example of the **bak lsdumps** command.

Viewing Recent Backup Information

Issue the **bak dumpinfo** command to list information about specific backups.

```
$ bak dumpinfo [{-ndumps number | -id dumpID}] [-verbose]
```

- The **-ndumps** *number* option specifies the number of dumps about which information is to be displayed; information about the most recent number of dumps specified with this option is displayed. Use this option or use the **-id** option; omit both options to list information about the last 10 dumps.
- The **-id** *dumpID* option specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this option or use the **-ndumps** option; omit both options to list information about the last 10 dumps.
- The **-verbose** option includes a detailed list of information about the dump specified with the **-id** option. This option can be used only with the **-id** option.

The dump ID, parent ID, dump level, and other dump information are displayed about the indicated dumps. The following example shows information about the last three dumps:

```
$ bak dumpinfo -ndumps 3

    DumpID   parentID lvl     created     nt nfsets dump_name
--------------------------------------------------------------
 729293644  729289323   1  02/09/93  5:34  1     43 users.tue
 729287531  729286818   1  02/08/93  4:52  1     23 users.mon
 729286056          0   0  02/07/93  4:27  1     31 users.wk1
```

Listing Tape Coordinator TCIDs

Issue the **bak lshosts** command to list all of the Tape Coordinators that have entries in the Backup Database.

```
$ bak lshosts
```

The output lists the name of the machine for which each Tape Coordinator is defined and the TCID of the Tape Coordinator. A Tape Coordinator's presence in the output does not imply that it is currently running.

```
$ bak lshosts

Tape hosts:
Host /.../abc.com/hosts/bak1, 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
```

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 specifies 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    tapename
654972910  654946323 1  10/01/91  5:07  10/01/91 4:01 users.tuesday.1
654960415  654946323 1  09/30/91  5:11  09/30/91 4:16 users.monday.1
654946323          0 0  09/29/91  5:36  09/28/91 4:31 users.week.1
```

Scanning the Contents of a Dump Tape

Important Note to Users

Scanning the contents of a dump tape is not applicable on OS/390 DFS.

To scan the contents of a dump tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See "Starting a Tape Coordinator" on page 299 for information on using the **butc** command to start a Tape Coordinator.)
2. Issue the **bak scantape** command to display information from a tape.

   ```
   $ bak scantape [-dbadd] [-tcid tc_number]
   ```

   The **-dbadd** option indicates that information extracted from the tape is to be added to the Backup Database; information is not added if the tape is damaged or if the entry has a dump ID number that is already used by an entry in the Backup Database. (See "Recovering Specific Backup Data" on page 314 for information about using this option.)
3. Place the tape in the drive, and press <Enter> in the corresponding Tape Coordinator's monitoring window. When using this command, you must insert tapes sequentially.

   An example of the output from this command, which lists tape label and fileset information, follows. The output is displayed in the monitoring window of the Tape Coordinator:

   ```
   $ bak scantape

   Tape label
   ----------
   name =         guests.monthly.1
   createTime =   Fri Nov 22 05:59:31 1990
   cell =         /.../abc.com
   size =         20103324 Kbytes
   dump path =    /monthly
   dump id =      729369701
   useCount =     1
   -- End of tape label --
   ```

```
-- fileset --
fileset name: user.guest10.backup
fileset ID 0,,112262
dumpSetName: guests.monthly
dumpID 729369701
level 0
parentID 0
endTime 0
clonedate Fri Nov 22 05:36:29 1991
```

Backing Up Data

The **bak dump** command is used to perform a dump operation. When you enter the command, specify the fileset family to be dumped and the level 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 data in all filesets included in the specified family is dumped; if you specify an incremental dump, only data in the filesets that has changed since they 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 period 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 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*. (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.

Attention: If a dump operation is interrupted or fails for any reason, you cannot be sure that any fileset is complete on one tape; the Backup Database contains an entry for the incomplete dump set, which cannot be used to restore data. Immediately restart the backup, using the same tape to record the dump set; using the same tape automatically removes the entry for the incomplete dump set from the Backup Database. If you use a different tape, you will need to use the **bak deletedump** command to manually remove the entry for the incomplete dump set from the database. (See "Deleting Backup Information" on page 307 for more information about the **bak deletedump** command.)

Using Tapes with a Backup Operation

Important Note to Users

On OS/390 DFS, standard label tapes are used for all dumps. Standard labels are generated by the system. The label referred to in this documentation isn't written to the tape. Instead it is written to a VSAM key sequenced data set (ksds). This information is used to maintain the records in the backup data base based on the expiration date specified for the dump level.

The Tape Coordinator dynamically allocates a tape drive for each dump set. Each dump set can be one or more tape data sets. The size of a tape data set is based on the **_IOE_BACKUP_TAPE_CAPACITY** environment variable value in the **BUTC***nn* process. Each tape data set can be a partial tape volume, a full tape volume or multiple tape volumes depending on how well the **_IOE_BACKUP_TAPE_CAPACITY** value matches the actual tape volume size. After **_IOE_BACKUP_TAPE_CAPACITY** bytes have been written (leaving room for header and trailer information), a new tape data set (and tape volume) will be started if the dump set has not been completed. A fileset can cross tape data sets.

The data set is cataloged in the OS/390 catalog. A data set can span multiple volumes. The data set name is derived from a predefined high level qualifier, the dump ID, and a sequence number.

Since the Tape Coordinator dynamically allocates a tape drive for each dump set, full and incremental dumps do not have to be readable by the same tape drive because the system will assign a tape drive based on the information stored in the catalog.

For more information on how to create the VSAM key sequence data set, see *OS/390 Distributed File Service DFS Configuring and Getting Started*.

Important Note to Users

The **TapeConfig** file described in this section is not required by the Backup Tape Coordinator on OS/390.

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. Any one fileset of the dumpset can span no more than five tape volumes.

Prior to performing a backup, you can preview the effects of your command without having the system actually perform the dump. Simply include the **-noaction** option with the **bak dump** command, specifying the remaining options as you would to really execute the dump. This lets you check a fileset family's size before actually dumping it so that you can calculate the correct number of tapes needed.

Backing Up a Fileset (Creating a Dump Set)

To back up a fileset, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 299 for information on using the **butc** command to start a Tape Coordinator.)
3. ***Optional:*** At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 300 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. 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.
6. 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).

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.

Important Note to Users

On OS/390 DFS, the Backup System automatically removes the record of a dump set from the Backup Database when the tape containing the dump set has expired.

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 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 “Recovering Specific Backup Data” on page 314 for more information about the **bak scantape** command.)

To delete backup information from the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. ***Optional:*** At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 300 for the advantages of interactive mode.) The commands in the following steps assume that regular command mode is used, not interactive mode.
3. Issue the **bak dumpinfo** command to list information, including dump IDs, about dump sets recorded in the Backup Database. A dump set's dump ID is required to delete it from the Backup Database.

   ```
   $ bak dumpinfo [{-ndumps number | -id dumpID}] [-verbose]
   ```

   - The **-ndumps** *number* option specifies the number of dumps about which information is to be displayed; information about the most recent **-ndumps** is displayed. Use this option or use the **-id** option; omit both options to list information about the last 10 dumps.
   - The **-id** *dumpID* option specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this option or use the **-ndumps** option; omit both options to list information about the last 10 dumps.
   - The **-verbose** option includes a detailed list of information about the dump specified with the **-id** option. The **-verbose** option can be used only with the **-id** option.
4. Issue the **bak deletedump** command to delete the desired dump set.

   ```
   $ bak deletedump -id dumpID
   ```

   The **-id** *dumpID* option specifies the dump ID number of the dump set whose entry is to be deleted from the Backup Database.

Restoring Data

When you restore data to the file system, you can restore individual filesets with the **bak restoreft** command, you can restore an entire aggregate with the **bak restoredisk** command, or you can restore all filesets for a family with the **bak restoreftfamily** command. Using the **bak restoreft** command, you can perform a full restore of the most recently dumped version of a fileset, or you can perform a date-specific restore, which restores the fileset to the state it was in at its last dump before the indicated date.

Data can be dumped and restored between different types of file systems. For example, data dumped from a DCE Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset. Similarly, data dumped from a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System fileset. You cannot dump or restore an OS/390 non-Local File System fileset.

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 Local File System fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Before performing a restore, you can direct the program 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, include the **-noaction** option with the **bak restoredisk**, **bak restoreft**, or **bak restoreftfamilty** command, along with any other options that you intend to use with the command.

Attention: If a restore operation is interrupted or fails for any reason, you cannot be sure that any fileset is complete in the file system; immediately restart the operation. If you do not, the file system may contain inconsistent information, which can result in problems in the future.

Specifying the Type and Destination of a Restore Operation

The system performs a full restore by default, recreating a fileset as it existed when it was last dumped (including data from the last full dump and any subsequent incremental dumps). You can direct the system to perform a date-specific restore by using the **-date** option with the **bak restoreft** command. This re-creates the fileset as it was when it was last dumped before the indicated date; it includes the last full dump and any incremental dumps done before the indicated date.

Using either the **bak restoreft**, **bak restoreftfamilty**, or **bak restoredisk** command, you can restore data to the same location that it previously occupied, in which case it overwrites the existing information, or you can restore it with a new name or at a location other than its original storage location. To overwrite the current contents of a fileset, use the **-server**, **-aggregate**, and **-fileset** options with the **bak restoreft** command.

To preserve the current contents of a fileset in the file system, use the **-extension** option to restore the data to a new fileset with the same name as the existing fileset by adding a distinguishing extension, such as **.restored**, with the **-extension** option. A new FLDB entry is created for the fileset and it is assigned its own fileset ID number. You can place the restored fileset at the same site as the existing fileset or at a different site by using the **-server** and **-aggregate** options.

To restore a fileset that no longer exists in the file system, simply provide the **-server**, **-aggregate**, and **-fileset** options. A new FLDB entry is created for the fileset and a fileset ID number is assigned.

Restoring Individual Filesets

Use the **bak restoreft** command to restore one or more individual filesets. Table 11 summarizes the options available with the **bak restoreft** command. Unless indicated as “Optional” in the table, each option is required.

Table 11. 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	If you dumped the **.backup** version of a fileset, add the **.backup** extension to the name you specify.
-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 period to separate the extension from each name, specify the period 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* indicate *hh:mm* as the time on day *mm/dd/yy*. A time must be in 24-hour format (for example, **20:30** for 8:30 p.m.).

To restore individual filesets, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 299 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. ***Optional:*** At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 300 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, using the information in Table 11 on page 309 as a guide:

   ```
   $ 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 the fileset. Specify the File Server machine using the machine's DCE pathname, the machine's hostname, or the machine's IP address.
   - The **-aggregate** *name* option is the device name or aggregate name of the aggregate to which each fileset is to be restored.
   - The **-fileset** *name* option is the name of each fileset to be restored.
   - The **-extension** *name_extension* option is the new extension to be added to 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 entries:

mm/dd/yy	Causes a date-specific restore of dumps that were done before 00:00 (12:00 a.m.) on the indicated date.
mm/dd/yy hh:mm	Causes a date-specific restore of dumps that were done before the specified time on the indicated date. The time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). Surround the entire argument with "" (double quotes) because it contains a space.

   - The **-noaction** option specifies that the command is to display the list of tapes needed to complete the restore without performing the actual operation.

Restoring a Fileset Family

Use the **bak restoreftfamily** command to restore a fileset or one or more specified filesets. Table 12 on page 311 summarizes the options available with the command. Unless indicated as “Optional” in the table, each option is required.

Table 12. Options Available with the bak restoreftfamily Command

Option	Specifies	Additional Information
-family	The fileset family to be restored.	Lets you restore all the filesets included in the fileset entries in a specified fileset family. If you use the **-family** option, you cannot use the **-file** option.
-file	Specifies the full pathname of a file from which the command uses to read the name of each fileset to be restored and the site to which it is to be restored to.	Us the **-file** option instead of the **-family** option to restore specific individual filesets. Each line of this file contains information for each fileset you want to restore. If you use the **-file** option, you cannot use the **-family** option.

To restore all the filesets included in a fileset entry, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 299 for information on using the **butc** process to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administration 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. The command in the following step assumes that the regular command mode is used, not interactive command mode.
4. Issue the **bak restoreftfamily** command with the appropriate options.

   ```
   $ bak restoreftfamily -family good.data
   ```

   - The **-family** option names the fileset to be retsored. All filesets contained within the fileset family **good.data** are restored to the sites recorded in their entries in the FLDB.

To restore specific filesets included in a fileset entry, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 299 for information on using the **butc** process to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administration 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. The command in the following step assumes that the regular command mode is used, not interactive command mode.
4. Create a file containing lines for each of the filesets you want to restore. Each line in the file is a specific file set:

   ```
   machine aggregate fileset [comments]
   ```

   For this example, the file **list.o.filesets** contains the following:

   ```
   /.../abc.com/hosts/dcedfs1 aggr1 fileset_x
   /.../abc.com/hosts/dcedfs1 aggr1 fileset_y
   /.../abc.com/hosts/dcedfs1 aggr2 fileset_z
   ```

   ```
   $ bak restoreftfamily -file /tmp/restore/list.o.filesets
   ```

   - The **-file** option names the full path of where the fileset restore information is found. Only those filesets contained within the file, **list.o.filesets**, are restored to the sites noted by the aggregate parameter. If you restore a fileset to the site where it currently exists, the command overwrites the existing version. If you restore to a site other than the site at which it currently exists, you must use the **fts zap** or **fts delete** command prior to the **restoreftfamily** command.

Restoring an Aggregate

Use the **bak restoredisk** command to restore all of the filesets on an aggregate. Table 13 on page 312 summarizes the options available with the command. Unless indicated as “Optional” in the table, each option is required.

Table 13. 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 files to an aggregate, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See “Starting a Tape Coordinator” on page 299 for information on using the **butc** process to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. ***Optional:*** At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See “Using the Interactive Interface” on page 300 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 you want to restore. Specify the File Server machine using the machine's DCE pathname, the machine's hostname, 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. This is necessary only if it is different from **-server**. Specify the File Server machine using the machine's DCE pathname, the machine's hostname, or the machine's IP address.
   - The **-newaggregate** *name* option is the device name or aggregate name of the aggregate on which the restored aggregate is to be placed. This is necessary only if it is different from **-aggregate**.
   - The **-noaction** option specifies that the command is to display the list of tapes needed to restore the aggregate without performing the actual operation.

Administering the Backup Database

A copy of the Backup Database can be installed on any server machine in a cell. The Backup Database stores two types of records used to track all of the backups done in the cell:

- Dump set records, which list the fileset family and the tape used in each dump set; and
- Administrative records, which list the fileset families, dump levels, and tape hosts.

Because information about dumps is difficult to re-create, 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 housing 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 **/opt/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. Re-create 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 re-created as well.

If specific information about a dump set is accidentally deleted from the Backup Database, you can use the **bak scantape** command with the **-dbadd** option to check the backup tape used for the dump set, recover the dump set information, and add the information to the Backup Database. Do *not* use the **bak scantape** command to attempt to reconstruct the database.

Backing Up the Backup Database

To back up the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See "Starting a Tape Coordinator" on page 299 for information on using the **butc** process 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.

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 WRITE (**w**) and EXECUTE (**x**) permissions on the **/opt/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 **/opt/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 “Starting a Tape Coordinator” on page 299 for information on using the **butc** process 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.

Recovering Specific Backup Data

Important Note to Users

The **bak scantape** command is not applicable against OS/390 tapes used by the OS/390 backup Tape Coordinator in OS/390 DFS.

Use the **bak scantape** command to extract dump set information from a backup tape and add it to the Backup Database. When you issue this command, you must place the backup tapes into the drive in sequential order. The system verifies that each tape is undamaged by checking the end-of-file markers that the Backup System inserts at the beginning and end of each fileset. If the markers are missing, the tape is assumed to be damaged and cannot be used for recovering data. To add information to the database, the entire tape must be undamaged, and the Backup Database must not contain an entry with the same dump ID as an entry being added.

To add recovered data to the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.

2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See "Starting a Tape Coordinator" on page 299 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.

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. Canceling a dump or restore operation while it executes can produce inconsistencies on a backup tape or in the file system.

If you cancel a backup operation while it is executing, all filesets written to tape before the **kill** signal is received are complete and usable on the tape. However, filesets being written when the signal is received may be incomplete and should not be used.

If you cancel a restore operation while it is executing, all completely restored filesets are online and usable. However, because most restore operations require data from multiple tapes (a full dump tape and one or more incremental dump tapes), most filesets are usually not completely restored. If the **kill** signal occurs before the system accesses all of the necessary tapes, most filesets are not restored to the desired date or version and should not be used.

If the interrupted restore operation is overwriting one or more existing filesets, the filesets can be lost entirely; however, the data being restored still exists on tape. In general, to avoid the inconsistencies that can result from an interrupted restore operation, reinitiate the restore operation.

Displaying Operations in Interactive Mode

Issue the **bak jobs** command to determine the job ID number of an operation. For an operation to appear in the output from the **bak jobs** command, you must have initiated the operation in interactive mode, and you must still be in interactive mode. No privileges are required to display an operation with the **bak jobs** command. (See "Using the Interactive Interface" on page 300 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`*`)` 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` for a restore operation initiated with the **bak restoreft**, **bak restoreftfamily**, or **bak restoredisk** command.
  - `Labeltape (`*`tape_label`*`)` 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` for a tape scanning operation initiated with the **bak scantape** or **bak readlabel** command.
  - `SaveDb` for a **bak savedb** operation.
  - `RestoreDb` 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 if displayed if the operation is executing normally.
  - The `[abort request]` message means the **bak kill** command was issued but the operation is not yet cancelled.
  - The `[abort request rcvd]` message means the **bak status** command was issued and the job is not yet cancelled.
  - The `[abort sent]` message means the operation is cancelled; 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 `[operator wait]` message is displayed if the Tape Coordinator is waiting for the operator to respond to an operator console reply prompt.
  - The `[drive wait]` message is displayed if the Tape Coordinator is waiting for the operator to insert a tape in the drive or waiting for a drive to be free.

Canceling Operations in Interactive Mode

For an operation to appear in the output from the **bak jobs** command, you must have initiated the command from interactive mode, and you must still be in interactive mode. The following commands assume you are in interactive mode (see “Using the Interactive Interface” on page 300). No privileges are required to list or kill a command, only to initiate one.

1. Issue the **bak jobs** command to determine the job ID number of the operation.

   ```
   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 id displayed for each job:
  - `Dump (`*`dump_set`*`)` 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` for a restore operation initiated with the **bak restoreft**, **bak restoreftfamily**, or **bak restoredisk** command.
  - `Labeltape (`*`tape_label`*`)` for a tape labeling operation started with the **bak labeltape** command (*tape_label* is the label being placed on the tape).
  - `Scantape` for a tape scanning operation initiated with the **bak scantape** or **bak readlabel** command.
  - `SaveDb` for a database saving operation initiated with the **bak savedb** command.
  - `RestoreDb` 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.
- One of the following messages reporting additional status information about the operation. No message is displayed if the operation is proceeding normally.
  - The `[abort request]` message is displayed if the **bak kill** command was issued but the operation is not yet cancelled.
  - The `[abort sent]` message is displayed if the operation is cancelled; 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 is displayed if the **bak** program temporarily loses contact with the Tape Coordinator executing the operation.
  - The `[drive wait]` message is displayed when the operation is waiting for the specified tape drive to become free.
  - The `[operator wait]` message is displayed if the Tape Coordinator is waiting for the operator to insert a tape in the drive.

2. Use the **bak kill** command to cancel the operation.

```
bak> kill {jobID | dump_set}
```

The *jobID* option is the unique job ID number of the operation to cancel; *dump_set* is the name of the operation in the form *fileset_family_name.dump_level* if it is a dump.

Note: The *jobID* option is the recommended operation for cancelling. If you use the *dump_set* option there is the possibility of having several instances of the same *dump_set* with unique job numbers assigned to each one. If you issue **kill** *dump_set* the first instance of the specified *dump_set* is canceled, the others are unaffected.

Chapter 16. Monitoring and Tracing Tools

This chapter describes how the **scout** program is used to monitor the File Exporter process on any File Server machine.

Monitoring File Exporters with the Scout Program

Important Note to Users

Scout does not highlight with reverse video. Instead, **scout** places a greater than sign (>) to the right of the highlighted field.

Cursor addressing does not apply to OS/390 DFS. On OS/390 DFS, after **scout** information is displayed, it is not refreshed. To see updated data, you must reenter the **scout** command.

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; 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 out the other fields for the machine. When the File Exporter returns to service, the machine name and the statistics are again displayed normally.

An Overview of the Scout Program

You can run **scout** from any machine configured as a DFS client or server. In a standard configuration, the binary file for **scout** is stored in **/opt/dfsglobal/bin/scout**. **Scout** can be run on OS/390, in TSO, or in the OS/390 shell environment.

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.

In most cells, all File Server machine names share the same basename, or common DCE prefix. For example, File Server machines in the **abc.com** cell have DCE pathnames like **/.../abc.com/hosts/fs1**, **/.../abc.com/hosts/fs2**, and so on. If all of the machines in a **scout** window have the same DCE pathname prefix, you can use the **-basename** option with the **scout** command. You can then specify the unique part of each machine name (**fs1** or **fs2**) and specify the basename (**/.../abc.com/hosts**) only once. You do not have to specify the / (slash) that separates the prefix from the unique part of each machine name; it is included automatically with the **-basename** option.

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 **scout**.
  - The **-basename** option displays the common DCE pathname prefix of the 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 connections with principals. This column displays the number of RPC connections open between the File Exporter and client machines. This number should equal or exceed the number in the `Ws` column, which shows the number of active client machines. The number in the `Conn` column can exceed the `Ws` number 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 the 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 the 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 contains the number of client machines (typically workstations) 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 available kilobyte blocks on each DFS aggregate on the File Server machine. For example, a display 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 entries, **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. In OS/390 DFS **scout** probes only once.

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 the File Exporter can have open to client machines 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 from the File Exporter 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 all exported aggregates and partitions on a File Server machine being monitored. Legal thresholds are integers from 0 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 all exported aggregates and partitions 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 any 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 exceed 1 megabyte (1024 kilobytes):

```
$ scout -server /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 -attention fetch 16
```

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. You can stop **scout** by entering the **exit** command.

Starting the Scout Program:

To start the **scout** program, do the following:

Open a command shell window for each instance of **scout** you want to run.

Initialize **scout** in each window (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 names 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 **-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 common to the File Server machines specified with **-server**. Do not include the / (slash) that separates the prefix from the unique part of each machine; it is included automatically with the **-basename** option. If the basename is specified, it is displayed in the banner line.

  Use this option only if you are specifying the DCE pathname or 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 **scout** in the banner line; this is useful if you are logged in to the machine remotely.
- The **-frequency** *seconds* option indicates how often **scout** is to probe the File Exporters. Specify a positive integer as a value in seconds; the default is 60 seconds. On OS/390 DFS, the **-frequency** option is ignored. On OS/390 DFS, **scout** probes the File Exporter once.

- The **-attention** *stat/threshold_pair* option specifies a list of attention settings (statistic and threshold pairs); **scout** highlights any value for a statistic that exceeds its threshold. (See “Setting Attention Thresholds” on page 321 for a discussion of legal values for this argument.)
- The **-debug** *filename* option enables debugging output and directs it to the specified *filename*. *Filename* can be an OS/390 data set or an HFS file.

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 and prints debugging information to the **scout.debug** file in the current directory.

```
$ scout -server fs1 fs2 -basename /.../abc.com/hosts -debug
scout.debug
```

The following example again instructs **scout** to monitor the File Exporters on the **fs1** and **fs2** machines in the **abc.com** cell; 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 more than 1 megabyte of data is fetched from a File Exporter; **scout** highlights an entry in the `Store` column if more than 512 kilobytes of data are stored by a File Exporter.

```
$ scout -server fs1 fs2 -b /.../abc.com/hosts -host -attention fetch 16 store 8
```

Stopping the Scout Program: The **scout** program does not run continuously. It probes the File Exporters only once, and then prints the debugging information. The probe of the File Exporters may take some time. You may wish to stop **scout** before it has finished probing.

To stop the **scout** program, enter the **exit** command from the command line where **scout** was started:

```
exit
```

Part 2. OS/390 DFS Reference

This part of the book discusses the reference information, organized into the following chapters.

- Chapter 17, "OS/390 System Commands" on page 327 provides information regarding the MVS system commands, **MODIFY**, **START**, and **STOP**.
- Chapter 18, "Distributed File Service Files" on page 337 introduces the **DFS** files for OS/390 DFS.
- Chapter 19, "Distributed File Service Commands" on page 401 provides descriptions of **DFS** command syntax for OS/390 DFS.
- Chapter 20, "Distributed File Service Application Programming Interface" on page 771 provides information on the application programming interface.

Each chapter begins with a short introduction and is followed by information about the daemons, the control programs, the control program commands, which are ordered alphabetically, and finally the application programming interface.

In addition, there are the following appendices.

- Appendix A, "Environment Variables in DFS" on page 775.
- Appendix B, "Sharing Data Between OS/390 UNIX Applications, Commands, and DFS Clients" on page 795.
- Appendix C, "Examples of Working with Character Data from Other Platforms" on page 799.
- Appendix D, "DFS/NFS Secure Gateway" on page 805.
- Appendix E, "Using Both SMB and DCE DFS" on page 807.

Chapter 17. OS/390 System Commands

This chapter introduces you to the following commands:

- **MODIFY**, an OS/390 system command which enables you to: start, stop, and query the status of DFS daemons.
- **START** and **STOP**, OS/390 system commands that enable you to start and stop the DFS Control Task (**DFS**).

These commands may be invoked from the operator console or from an OS/390 Spool Display and Search Facility (SDSF) screen.

modify dfs daemon

Purpose

Starts, stops, or queries the status of DFS Control Task daemons. This command can also be used to send a command string to the **boserver** or **dfskern** daemon.

Format

You can use any of the following formats for this command.

```
modify procname,{start | stop | query} daemon

modify procname,send boserver,query

modify procname,send dfskern,{clonesys,aggrID[,'prefix'] |
                              reload,idmap |
                              release,data-set-name |
                              release,high-level-qualifier}
```

Parameters

procname The name of the DFS Control Task. On OS/390 DFS, the default *procname* is **dfs**.

command The action that is to be performed on the DFS daemon or daemons. This parameter can have one of the following values:

- **start** Starts either a single DFS daemon or all the DFS daemons.
- **stop** Stops either a single DFS daemon or all the DFS daemons.
- **query** Displays the status and process identifier (PID) of the DFS daemon or DFS daemons.
- **send** Sends a command string to the **boserver** or **dfskern** daemon.

daemon The name of the OS/390 DFS Control Task daemon for which the action is being requested. This parameter can have one of the following values: .

- **boserver** The **Basic OverSeer Server** (BOS) daemon. The **boserver** daemon monitors all server processes that it starts on the local machine. The **boserver** restarts any failed processes automatically, limiting system outages. For information on processes started by the **boserver** daemon, see “bosserver” on page 538.

  The **boserver** daemon parameter of the **send** command can be further modified through the following parameter.

  - **query** Displays the status and process identifier (PID) of the **boserver** daemons.
- **butc***nn* The **BackUp Tape Coordinator** (BUTC) daemon. The **butc** server daemon is the tape coordinator machine on which backup and restore operations are physically conducted. Valid entries for *nn* are 01 through 08.
- **export** The **export** daemon. The **export** server daemon allows exporting of aggregates.
- **unexport** The **unexport** daemon. The **unexport** server daemon allows unexporting of aggregates.

dfskern The **dfskern** daemon. The **dfskern** server daemon is the DFS File Exporter process.

The **dfskern** daemon parameter of the **send** command can be further modified through the following parameters:

clonesys,*aggrID***[,'***prefix_string***']**
Creates backup versions of all indicated read-write DCE Local File System filesets. The *aggrID* is the aggregate ID of the OS/390 aggregate that contains the filesets to clone. If '*prefix_string*' is specified, only those filesets with a name that matches the *prefix_string* are cloned. Otherwise, all the filesets in the aggregate are cloned. If a backup version of a fileset already exists, the new clone replaces it. The DFS server principal for this machine must be in the **admin.fl** list.

This command should be used for OS/390 DCE Local File System aggregates when the **clonesys** function is desired and an OS/390 operator console or SDSF is available. It provides significant performance benefits by minimizing the number of RPCs required. Otherwise, the **fts clonesys** command may be used.

reload,idmap
Ensures that new identity mappings are used by first eliminating any previous mappings cached by the system. Then the parameter will, depending on how DCE principal names are mapped to OS/390 user IDs by your system, reload a new Identity Mapping Output file or use updated values supplied through the Resource Access Control Facility (RACF) identity mapping function.

For example, if your system is configured to use the Identity Mapping Output file method to map DCE principal names to OS/390 user IDs, issuing the **reload,idmap** parameter eliminates any previous identity mappings cached by the system. It then reloads a new Identity Mapping Output file. If the RACF identity mapping function is configured on your system, the **reload,idmap** parameter will eliminate previous identity mappings cached by the system, allowing the DFS server to use any updated RACF identity mappings.

The type of method used to map DCE principals to OS/390 user IDs is defined during system configuration by the **_IOE_MVS_IDMAP_SAF** environment variable. For further information, see “Mapping DCE User IDs to OS/390 User IDs” on page 174.

release,*data-set-name*
The *data-set-name* allows the operator to force the DFS File Server to free up an OS/390 data set if an important batch job has been waiting too long. The operator should use this command rather than stopping **dfskern** or DFS. (Normally, the DFS File Server should release an OS/390 data set that a batch job is waiting for immediately. See “Sharing Data” on page 237 for more information.)

release,*high-level-qualifier*
The *high-level-qualifier* allows the operator to force the DFS File Server to refresh the root directory file name and status cache. The operator should use this to refresh the DFS File Server cache of exported record data set names before the refresh interval time occurs. See the **_IOE_RFS_STATUS_REFRESH_TIME** environment variable on page 784, for more information.

all	All of the DFS daemons.

Important Note to Users

In DFS, there are two types of processes: **simple** and **cron**. A **simple** process is defined as a continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes. A **cron** process refers to a process that runs independently of any other processes; however, unlike a **simple** process, a **cron** process runs periodically, not continuously.

On OS/390 DFS, **simple** processes are not identified by complete pathnames to the binary files for the processes. All **simple** process commands on OS/390 DFS are members of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) created during installation (for further information, refer to the *OS/390 Program Directory*). As all OS/390 member names are limited to eight or less characters, the **bosserver** process member name is shortened to **boserver**.

The following are the PDS member names for the DFS processes started by the DFS Control Task:

boserver	The **boserver** daemon load module name. The name, **boserver**, is an alias for the load library entry, **IOEBOSRV**.
butc*nn*	The **butc***nn* daemon load module name. There are eight valid load module names for the butc processes: **butc01**, **butc02**, **butc03**, **butc04**, **butc05**, **butc06**, **butc07**, and **butc08**. All are valid aliases for the single load module used for all **butc***nn* processes- **IOEBUTC**.
dfskern	The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.

In addition, the **export** process is started by the DFS Control Task and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.

Usage

The **modify dfs daemon** command is used to manually start or stop one or more DFS daemons, or to view the status of the daemons. It is especially useful in situations where a daemon has stopped abnormally. On OS/390 DFS, the DFS daemons are contained in the **dfs** address space.

Starting and Stopping OS/390 DFS Daemons: Using the **MODIFY** system command, you can start or stop either a single daemon or all the daemons configured on the host. However, because the successful startup of some of the DFS daemons depends on the availability of the services provided by other DFS daemons, there is an order in which DFS daemons must be started. The **all** command starts the daemons in the correct order. See "Order of Starting DFS Server Daemons" on page 121 for more information about the starting order of the daemons.

Viewing the Status of OS/390 DFS Daemons: Using the **MODIFY** system command, you can view the status of the DFS daemons from the operator console using the *query* option.

Sending a Command String to the boserver Daemon: Using the **MODIFY** system command, you can send a command string to the **boserver** daemon. The **query** parameter displays the status of the **boserver** and its subprocesses (**ftserver**, **repserver**, **flserver**, **bakserver**, **upserver**, **upclient**) to the operator console.

Sending a Command String to the dfskern Daemon: Using the **MODIFY** system command, you can send a command string to the **dfskern** daemon. The **clonesys** parameter provides a subset of the function of the **fts clonesys** command. It allows an OS/390 console operator to request the clone function for an OS/390 resident aggregate. It provides significant performance benefits by minimizing the number of RPCs required but can only be used for OS/390 aggregates. The **reload,idmap** parameter eliminates any cached identity mappings and, depending on how the system is configured, either reloads a new Identity Mapping output file or uses RACF's identity mapping function to update identity mappings between DCE principal names and OS/390 user IDs.

Privilege Required

This command is an OS/390 system command.

The **clonesys** parameter requires that the DFS Server principal for this machine be contained in the **admin.fl** list.

Examples

The following example starts the DFS **boserver**:

```
modify dfs,start boserver
```

The following example starts all the DFS daemons:

```
modify dfs,start all
```

The following example views the status of all the DFS Control Task daemons that are currently active on the host.

```
modify dfs,query all
```

In the following example, on a system not using the RACF identity mapping method, the **dfskern** daemon is instructed to eliminate existing identity mappings and to reload an updated identity mapping output file that includes recently added or deleted user identity mapping information:

```
modify dfs,send dfskern,reload,idmap
```

Note: The command string above would be identical if the RACF identity mapping method was configured on your system. In this case, the **dfskern** daemon would be instructed to eliminate any existing identity mappings and to use any updated RACF identity mappings.

The following example creates backup versions of all the filesets in the OS/390 aggregate that has an aggregate ID of `103` and have a fileset name that begins with `'user'`.

```
modify dfs,send dfskern,clonesys,103,'user'
```

Related Information

Command:

bosserver	**dfsexport**	**fxd**
Butc		

File:
ioepdcf

start dfs

Purpose

Starts the DFS Control Task.

Format

start *procname*[,*start_options*]

Parameters

procname	The name of the DFS Control Task. On OS/390 DFS the default *procname* is **DFS**.
start_options	The value passed to the **START** command. On OS/390 DFS, *start_options* has only one value, *parm='-nodfs'*. You can use the *parm='-nodfs'* option to start the DFS Control Task without starting the DFS Control Task daemons.

Usage

The **START** command is used to initiate the DFS Control Task.

You can use the *parm='-nodfs'* option to start the DFS Control Task without starting the DFS Control Task daemons.

Note: In OS/390 DFS, *start_options* values **must** be enclosed in single quotes. This is because the OS/390 **START** command converts all user-supplied *start_options* values to uppercase characters.

Privilege Required

This command is an OS/390 system command.

Examples

The following command starts the DFS Control Task and all daemons on OS/390 DFS:

```
start dfs
```

The following command starts the DFS Control Task without starting the DFS daemons:

```
start dfs,parm='-nodfs'
```

Related Information

Command:

bosserver	**Butc**
dfsexport	
fxd	

stop dfs

Purpose

Stops the DFS Control Task processes and processes controlled by the **bosserver**.

Format

stop *procname*

Parameters

procname The name of the DFS Control Task. On OS/390 DFS the default *procname* is **DFS**.

Usage

The **STOP** command stops the DFS Control Task (**dfscntl**), the processes controlled by the **dfscntl** process, and processes controlled by the **bosserver** process. These processes are:

bosserver
The **Basic OverSeer Server** (BOS) daemon. The **bosserver** daemon monitors all server processes that it starts on the local machine.

Processes controlled by the **bosserver** daemon are:

bakserver
The **Backup Server** (bakserver) daemon. The **bakserver** communicates with the backup database to perform dump and restore operations (for further information, see "bakserver" on page 473).

flserver The **Fileset Location Server** (flserver) daemon. The **flserver** maintains the Fileset Location Database (FLDB), which tracks the location of all DCE Local File System and non-Local File System filesets (for further information, see "flserver" on page 605).

ftserver The **Fileset Server** (ftserver) daemon. The **ftserver** handles fileset administration operations such as creating, deleting, moving or replicating filesets (for further information, see "ftserver" on page 720).

repserver
The **Replication Server** (repserver) daemon. The **repserver** tracks the currency of replicas and updates the versions of data used at each replication site. (for further information, see "repserver" on page 745).

upserver The **Update Server** (upserver) daemon. The **upserver** distributes files such as common configuration files and administrative lists from System Control machines to other server machines in a domain (for further information, see "upserver" on page 768).

upclient The **Update Client** (upclient) daemon. The **upclient** checks specified files and directories on the local machine and ensures they match the corresponding files on the System Control machine (for further information, see "upclient" on page 765).

butc*nn* The **BackUp Tape Coordinator** (BUTC) daemon. The **butc** server daemon is the tape coordinator machine on which backup and restore operations are physically conducted. Valid entries for *nn* are 01 through 08.

export	The **export** daemon. The **export** server daemon allows exporting of fileset aggregates.
unexport	The **unexport** daemon. The **unexport** server daemon allows unexporting of fileset aggregates.
dfskern	The **dfskern** daemon. The **dfskern** server daemon is the DFS File Exporter process.
dfscntl	The **dfscntl** daemon. The **dfscntl** server daemon is the DFS control program.

Important Note to Users

In DFS, there are two types of processes: **simple** and **cron**. A **simple** process is defined as a continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes. A **cron** process refers to a process that runs independently of any other processes; however, unlike a **simple** process, a **cron** process runs periodically, not continuously.

On OS/390 DFS, **simple** processes are not identified by complete pathnames to the binary files for the processes. All **simple** process commands on OS/390 DFS are members of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) created during installation (for further information, refer to the *OS/390 Program Directory*). As all OS/390 member names are limited to eight or less characters, the **bakserver**, **bosserver**, and **repserver** process member names are shortened to **bkserver**, **boserver**, and **rpserver**, respectively. For additional information, see the examples on page 488.

The following are the PDS member names for the DFS processes controlled by the **bosserver** daemon:

bkserver	Starts the Backup Server process. The name, **bkserver**, is an alias for the load library entry, **IOEBKSRV**. For further information, see “bakserver” on page 473.
flserver	Starts the Fileset Location Server process. The name, **flserver**, is an alias for the load library entry, **IOEFLSRV**. For further information, see “flserver” on page 605.
ftserver	Starts the Fileset Server process. The name, **ftserver**, is an alias for the load library entry, **IOEFTSRV**. For further information, see “ftserver” on page 720.
upclient	Starts the Update Client process. The name, **upclient**, is an alias for the load library entry, **IOEUPCLN**. For further information, see “upclient” on page 765.
upserver	Starts the Update Server process. The name, **upserver**, is an alias for the load library entry, **IOEUPSRV**. For further information, see “upserver” on page 768.
rpserver	Starts the Replication Server process. The name, **rpserver**, is an alias for the load library entry, **IOERPSRV**. For further information, see “repserver” on page 745.

The following are the PDS member names for the DFS processes started by the DFS Control Task:

boserver	The **boserver** daemon load module name. The name, **boserver**, is an alias for the load library entry, **IOEBOSRV**.
butc*nn*	The **butc***nn* daemon load module name. There are eight valid load module names for the butc processes: **butc01**, **butc02**, **butc03**, **butc04**, **butc05**, **butc06**, **butc07**, and **butc08**. All are valid aliases for the single load module used for all **butc***nn* processes- **IOEBUTC**.
dfskern	The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.

In addition, the **export** process is started by the DFS Control Task and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.

Privilege Required

This command is an OS/390 system command.

Examples

The following command stops the DFS Control Task and all daemons on OS/390 DFS:

```
stop dfs
```

Related Information

Commands:

bosserver	**Butc**
dfsexport	
fxd	

Chapter 18. Distributed File Service Files

This chapter introduces you to the Distributed File Service files. In DFS, there are several system-specific files that contain important system information. These files contain cached data, information about cached data, and output from processes and commands. They also are used to define configuration parameters for server processes and to list the users, groups, and servers allowed to access the various server processes in DFS.

This chapter provides an alphabetical listing of all relevant DFS files.

DFS Files Overview

Purpose

Introduction to DFS files.

Usage

DFS includes a number of system-specific files. These files can be grouped into the following general categories:

Configuration files
Define configuration parameters for specific server processes such as a tape coordinator or Cache Manager.

Administrative lists
List the principals (users, groups, and servers) allowed to access specific server processes, including the Backup Server, the Basic OverSeer Server, the Fileset Server, the Fileset Location Server, and the Update Server.

Cache-related files
Contain cached data or information about cached data.

Log files Contain output from specific processes or commands.

Specific information about the files, such as pathnames and format, is included with the description of each of them. Most of the files are also referenced in Part 1, “OS/390 DFS Administration Guide.” Refer to that part of this book for more information about these files and the DFS components and commands with which they are associated.

Implementation Specifics

These files are part of the OS/390 DFS base.

On OS/390 DFS, most commands and files can be run from Time Sharing Option Extensions (TSO/E), submitted as a batch job, or run from the OS/390 shell. For a complete list of commands and the corresponding names, if available, for running them in TSO/E, batch and the OS/390 shell, see Table 17 on page 405. For simplicity, the commands, files and examples used in this book are shown as entered from the OS/390 shell. All files are stored in Extended Binary Coded Decimal Interchange Code (EBCDIC) on OS/390 DFS.

Changes to commands or files for OS/390 DFS will be identified in the **Implementation Specifics** section for the commands or files.

Related Information

Following is a list of all relevant DFS files that are described in this section.

All DFS system-specific files are listed for completeness. Files not supported on OS/390 DFS are clearly identified later in this book.

See Chapter 19, “Distributed File Service Commands” on page 401 for information about any of the commands referenced in the following sections.

- Configuration files:

Attributes (rfstab)	**devtab**	**NoAuth**
BosConfig	**dfstab**	**TapeConfig**
CacheInfo	**ioepdcf**	

- Administrative files:

admin.bak	**admin.fl**	**admin.up**
admin.bos	**admin.ft**	

- Cache-related files:

CacheItems	**FilesetItems**	**Vn**

- Log files:

BakLog	**FlLog**	**TE_***tapennn*
BosLog	**FMSLog**	**TL_***tapennn*
CMLog	**FtLog**	**UpLog**
DFSgwLog	**RepLog**	

Attributes File (rfstab)

Purpose

Contains tables describing the attributes used to manipulate record files in the DFS Server. It also contains descriptions for:

- Data set creation attributes
- Processing attributes
- Site attributes.

The client user can also modify data set creation attributes that provide information about an OS/390 file to the DFS Server, such as the type of file, or how the file is allocated (for example, blocks, cylinders, or tracks). Processing attributes and site attributes can only be modified by the system administrator.

Notes:

1. For information on attributes for the DFS client, refer to "Creating OS/390 Files" on page 241.
2. The Attributes file is sometimes referred to as the **rfstab** file.

Specifying the Location of the DFS Attributes File: The attributes file is in either an HFS file, a fixed-block partitioned data set, or a fixed-block sequential data set with a record length of 80. The file's location is specified by the **_IOE_RFS_ATTRIBUTES_FILE** environment variable for the **dfskern** process. For example, **`_IOE_RFS_ATTRIBUTES_FILE=/opt/dfslocal/var/dfs/rfstab`**, this is the default location.

If no attributes are specified for specific files, DFS can use the same attributes file as the DFSMS/MVS NFS Server. For example, **`_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'`**, would cause DFS to use the member NFSSATT in PDS NFSADMIN.NFSS for the DFS record file attributes. (The NFS Server attributes file is specified in the NFSATTR DD statement of the NFS Server startup procedure.) NFS Server attributes that are not supported by DFS are ignored. See "Unsupported Attributes" on page 347 for a list of attributes that are not supported.

A DFS attributes file can also be specified on an RFS fileset basis. It can be specified in the **devtab** entry for the RFS fileset on the same line as the data set name: **attrfile** *attributes_file* (where *attributes_file* is the name of the attributes file that controls the data set creation, processing and site attributes for this RFS fileset). For example,

```
* RFS devices
define_ufs 3 rfs
USERA attrfile /opt/dfslocal/var/dfs/rfstab2
```

If no **attrfile** parameter is specified in the **devtab** entry for the RFS fileset, the attributes are taken from the attributes file specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable in the **dfskern** process.

If this is not specified by the **_IOE_RFS_ATTRIBUTES_FILE** environment variable and if **/opt/dfslocal/var/dfs/rfstab** does not exist then the DFS Server system defaults are used for RFS attributes.

Specifying Attributes for a Specific File: Within a DFS attributes file, file attributes (creation and processing) can be specified on an individual file basis. The attributes file is first scanned for specific file attributes. This is indicated at the beginning of the attributes file by the **startfileattributes:** and **endfileattributes:** keyword pair. Within these two keywords, there may be multiple **file:** keywords. Each **file:** keyword is followed by one or more filenames. The filename specified is the OS/390 data set name minus the exported prefix. The attributes for this file or files are specified on subsequent lines and delimited by another **file:** keyword or the **endfileattributes:** keyword. If a file being accessed is not found in the specific file attributes section of the attributes file, the attributes are taken from the general section of the attributes file.

Using Multipliers: Instead of entering the entire numeric values for the attributes, you can use the multipliers K (1024), M (1024 × 1024), or G (1024 × 1024 × 1024) for specifying sizes. For example, lrecl (8192) is the same as lrecl (8K).

Data Set Creation Attributes: The data set creation attributes are used to define the structure of OS/390 data sets when creating a file. These attributes correspond to the data control block (DCB) or the job control language (JCL) parameters used to define an OS/390 data set when it is created. Refer to the *OS/390 MVS JCL Reference*, GC28-1757, for more detailed information about the data set creation attributes.

The data set creation attributes are described in Table 14. Defaults are underlined.

You can override these attributes by specifying an attributes file in the **devtab** entry for the fileset or by using a file creation command. For PDS and PDSE, members have the same attributes as the data set attributes, so the file creation attributes specified for members are ignored.

Table 14 (Page 1 of 3). Data Set Creation Attributes

Data Set Creation Attribute	Description
<u>**blks**</u>	Specifies that disk space (see the **space** attribute in this table) is allocated by blocks, except for VSAM data sets.
cyls	Specifies that disk space is allocated by cylinders.
recs	Specifies that disk space is allocated by records for VSAM data sets. **blks** and **recs** are identical for VSAM data sets.
trks	Specifies that disk space is allocated by tracks.
blksize(<u>0</u> | *quan*)	Specifies the maximum length, in bytes, of a physical block on disk. *quan* is a number from <u>**0**</u> (the default) to 32,760. If blksize(0) is specified, the system determines an optimal block size to use.
dataclas(*class_name*)	Specifies the data class associated with the file creation. The *class_name* must be defined to DFSMS/MVS before it can be used by the client. The system-managed storage automatic class selection routine must also assign a storage class to the file being created. For more information on data classes, refer to the *DFSMS/MVS Version 1 Release 3, DFSMSdfp Storage Administration Reference*, SC26-4920.
dir(<u>27</u> | *quan*)	Specifies the number of 256-byte records needed in the directory of a PDS. Use it with the **mkdir** command when you are creating a PDS. *quan* is a number from 1 to 16,777,215 (the default is <u>**27**</u>). The maximum number of PDS members is 14,562.
dsntype(library | <u>pds</u>)	Specifies whether a PDSE or a PDS is to be created when the **mkdir** client command is used. **library** is for PDSE. **pds** is for PDS. You cannot create a PDS (or PDSE) within another PDS (or PDSE). If you need help deciding whether to create a PDS or a PDSE, refer to the *DFSMS/MVS Version 1 Release 3, Using Data Sets* book, SC26-4922.

Table 14 (Page 2 of 3). Data Set Creation Attributes

Data Set Creation Attribute	Description
dsorg(*org***)**	Specifies the organization of a data set. *org* can be a physical sequential (**ps**) data set, direct access (**DA**) data set, VSAM KSDS (**indexed**), VSAM RRDS (**numbered**) or VSAM ESDS (**nonindexed**). This attribute is ignored for directory-oriented client commands. If you are using VSAM data sets in binary mode with an AIX client, then **nonindexed** is recommended.
keys(*len, off***)**	Specifies the length and offset of the keys for VSAM KSDS data sets. Keys can only be specified when using **dsorg(indexed)**. *len* and *off* are specified in bytes. *len* is between 1 and 255 (the default is **64**). *off* is between 0 and 32,760 (the default is **0**). When you create a VSAM KSDS data set, the records you are loading into it must be keyed-sequenced or the write fails. Each write of the data set is treated like a first load, and requires that the records being loaded are in ascending key sequence.
lrecl(8196 | *quan***)**	Specifies: 1. The length, in bytes, for fixed-length records. 2. The maximum length, in bytes, for variable-length records. If the **blksize** attribute is specified, the value must be at least 4 bytes less than the **blksize** quantity. *quan* is a number from 1 to 32,760 (the default is **8196**).
mgmtclas(*mgmt_class_name***)**	Specifies the management class associated with the file creation. The *mgmt_class_name* must be defined to DFSMS/MVS before it can be used by the client. The system managed storage automatic class selection (ACS) routine must also assign a storage class to the file being created. For more information on management classes, refer to the *DFSMS/MVS Version 1 Release 3, DFSMSdfp Storage Administration Reference*, SC26-4920.
model(*dsname***)**	The name of the cataloged VSAM data set from which to copy data set creation attributes when creating a new VSAM data set. *dsname* is a fully-qualified MVS data set name without quotation marks. The **model** attribute must be used with one of the **dsorg** attributes which imply a VSAM organization. You can do this by specifying the **dsorg** attribute for a VSAM in the command. See the **dsorg** entry in this table.

recfm(*cccc***)**	Specifies the format and characteristics of the records in the data set. *cccc* can be 1 to 4 characters, in one of the following combinations:
	`f | fb | fs | fbs`
	`u`
	`v | `**`vb`**` | vs | vbs`
	Valid record format characters:
	b Blocked
	f Fixed-length records
	s Spanned for variable records, standard format for fixed records
	u Undefined-length records
	v Variable-length records
	In **recfm**, codes **v**, **f** and **u** are mutually exclusive. The **s** code is not allowed for a PDS or PDSE.

recordsize(*avg,max***)**	The average and maximum record size for VSAM data sets. *avg* and *max* are specified in bytes. They can each range from 1 to 32,760 (the defaults are **512** and **4096**, respectively). These values must be equal for VSAM RRDS.

Table 14 (Page 3 of 3). Data Set Creation Attributes

Data Set Creation Attribute	Description
rlse	Specifies that unused space should be released from the data set the first time a new data set is closed. For slow clients with long pauses between writes, the **rlse** attribute causes space to be released from the primary extent prematurely. Further writes cause secondary space to be allocated.
<u>norlse</u>	Specifies that unused space should not be released from the data set.
shareoptions(*xreg,xsys***)**	Specifies the cross-region and cross-system share options for a VSAM data set. *xreg* is a number from 1 to 4; *xsys* is either 3 or 4. The defaults are **<u>1</u>** and **<u>3</u>**, respectively.
	This applies to VSAM data sets only. For spanned records of non-VSAM data sets, see the entry for **recfm** in this table.
spanned	Specifies that VSAM KSDS or ESDS data sets can contain records that span control intervals (spanned records).
<u>nonspanned</u>	Specifies that data sets do not have spanned records.
space(*prim[,aux]***)**	Specifies the amount of primary and auxiliary space allocated for a new data set on a direct access volume. *prim* is the number (from 0 to 16,777,215) of primary tracks, cylinders, or data blocks in the data set. *aux* (optional) is the number (from 0 to 16,777,215) of additional tracks, cylinders, or blocks allocated if more space is needed. If this attribute is not specified, the default is used. The defaults are **<u>100</u>** and **<u>10</u>**, respectively.
storclas(*class_name***)**	Specifies the storage class associated with the file creation. The *class_name* must be defined to the DFSMS/MVS before it can be used by the client. For more information on storage classes, refer to the *DFSMS/MVS Version 1 Release 3, DFSMSdfp Storage Administration Reference*, SC26-4920.
unit(*unit_name***)**	Specifies the unit on which to create a data set. *unit_name* is a generic or symbolic name of a group of DASD devices. The *unit_name* must be specified as 3390 for extended format data sets. **Note:** You cannot create or access tape data sets on an OS/390 host using the DFS Server. You cannot create extended format data sets with the DFS Server, except by ACS routines.
vol(*volser***)**	Specifies the name of the DASD volume to be used to store the created data set. **vol** is the keyword and *volser* represents the volume name. If a data set is to be system-managed, as is determined by the DFSMS/MVS automatic class selection (ACS) routines, you can omit this attribute.

Processing Attributes: Processing attributes are used to control how files are accessed by clients. The processing attributes are described in Table 15 on page 345. Defaults are underlined. The administrator can override the default processing attributes in the **devtab** entry for the fileset. The client user cannot override processing attributes.

Table 15 (Page 1 of 2). Processing Attributes

Processing Attribute	Description
binary	Indicates that the data set is processed between the client and server using binary format and no data conversion occurs between ASCII and EBCDIC formats.
text	Converts the contents in the data set between EBCDIC and ASCII formats. Use this format to share text data between clients and OS/390 applications. In text mode, the following attributes apply: • **<u>blankstrip</u>** and **noblankstrip**. See the entry for **<u>blankstrip</u>** in this table. • End-of-line specifiers (**<u>lf</u>**, **cr**, **lfcr**, **crlf**, or **noeol**) are used to indicate the OS/390 logical record boundary. See the entry for **<u>lf</u>** in this table.
<u>blankstrip</u>	With text mode, strips trailing blanks at the end of each record of a fixed-length text file when the file is read. Pads the end of each file or record with blanks when a text file is written.
noblankstrip	Does not strip trailing blanks at the end of fixed-length records when a fixed-length text file is read. Does not pad records when writing a text file. The file must be of the correct size or an I/O error is reported to the client. For information on the **text** mode, see the **text** option of **devtab** (page 368).
	With text mode, use one of the following end-of-line specifiers:
<u>lf</u>	Line Feed is the end-of-line terminator (standard AIX or UNIX).
cr	Carriage Return is the end-of-line terminator.
lfcr	Line Feed followed by Carriage Return is the end-of-line terminator.
crlf	Carriage Return followed by Line Feed is the end-of-line terminator (standard DOS).
noeol	No end-of-line terminator. For information on the **text** mode, see the **text** option of **devtab** (page 368).
<u>executebiton</u>	Turns on the execute bits in user, group, and other (as reported with the **ls** (list) AIX or UNIX command) for files. Use when storing executables or shell scripts on the OS/390 system.
executebitoff	Turns off the execute bits in user, group, and other for the mount point's files. This value is normally used in the site file.
fastfilesize	Causes the DFS File Server to approximate the file size. For more information, see “Handling of the File Size Value” on page 246.
<u>nofastfilesize</u>	For Direct Access data sets (PDSs and PDSEs) and non-system managed data sets, this specifies to read the entire file or member to get the file size. Using this attribute might cause a noticeable delay when first accessing very large data sets. For more information, see “Using fastfilesize to Avoid Read-for-Size” on page 248.
<u>mapleaddot</u>	Turns on mapping of a single leading "." from a client file name to a legal leading "$" on OS/390. This option would normally be enabled for access by AIX and UNIX clients.
nomapleaddot	Turns off mapping of a single leading "." from a client to a leading "$" on OS/390.
<u>maplower</u>	Turns on mapping of lower case file names to upper case when accessing files on OS/390, and back when sending to the network. This option would normally be enabled for access by AIX or UNIX clients. This option only affects file names (high-level qualifiers and user catalog aliases).
nomaplower	Turns off mapping of lower case file names to upper case and back when using files on OS/390.

Table 15 (Page 2 of 2). Processing Attributes

Processing Attribute	**Description**
	The DFS Server uses DFSMShsm to recall or delete migrated files. The action that the server takes against the migrated files depends on which of the **retrieve** or **noretrieve** attributes is active.
retrieve(nowait)	When the **retrieve(nowait)** attribute is active, the server does not wait for the recall to finish, and immediately returns a "device not available" message. You can try accessing the file later when the recall has completed.
noretrieve	When the **noretrieve** attribute is active, the server does not recall the file, and can return "device not available" upon a lookup, an rdwr, or a create request for a file. For more information, see "Retrieve Attributes" on page 346, following this table.
setownerroot	Sets the user ID in a file's attributes to `root`.
setownernobody	Sets the user ID in a file's attributes to `nobody`.

Retrieve Attributes: The server deletes the migrated file upon a remove request for a file, regardless of whether the **retrieve** or the **noretrieve** attribute is active. Typically, a remove request is preceded by a lookup request. If the dataset was migrated with DFSMS/MVS 1.2 or below, retrieve attribute causes a recall because lookup processing needs to open the dataset and read for size. If the dataset was migrated under DFSMS/MVS 1.3 and DFSMShsm™ 1.3, and is SMS managed, its attributes were saved on DASD; therefore it is not always necessary to recall the data set to read for size and the dataset may be deleted without recall. If the **noretrieve** attribute is active, the lookup can return a "device not available" message. If the client code decides to ignore the error and go with the remove, the migrated file is then deleted.

The UNIX command **ls mvsusera** does not issue requests for individual files under the `mvsusera` directory. Migrated files under the `mvsusera` directory are displayed, but are not recalled. However, the UNIX command **ls -l mvsusera** issues lookup requests for individual files under the `mvsusera` directory.

Site Attributes: The site attributes are used to control DFS Server resources. These attributes are described in Table 16. Site attributes can be overridden in the **devtab** entry for the fileset.

Table 16 (Page 1 of 2). Site Attributes

Site Attribute	**Description**
bufhigh(*n*)	Specifies the maximum size (in bytes) of allocated buffers before buffer reclamation (see the **percentsteal** attribute in this table) is initiated. *n* is an integer from 1MB to 128MB (the default is **2MB**). If the combined total specified in the **bufhigh** and **logicalcache** attributes is greater than the available storage in the extended private area (implied by the REGION parameter in your procedure) at startup, the server shuts down immediately. A higher number means more caching and potentially better read performance.
cachewindow(*n*)	Specifies the window size used in logical I/O to buffer client block writes received out of order. *n* is a number from 1 to 256 (the default is **16**).
filetimeout(*n*)	Specifies the amount of time, in seconds, before a data set is closed and data is written to DASD. The minimum specification is 30. The default is 30.
logicalcache(*n*)	Specifies the maximum size (in bytes) of allocated buffers in the logical I/O processing for all the cache windows combined. *n* is an integer from 1MB to 128MB (the default is **1MB**).

Table 16 (Page 2 of 2). Site Attributes

Site Attribute	Description
maxrdforszleft(*n***)**	Defines the number of physical block buffers left after determining a file's size. This operation is done for later server read requests to the same file. The buffers left are subject to trimming during a "buffer steal" operation. *n* is an integer from 1 to 1024 (the default is **32**).
percentsteal(*n***)**	Specifies the percent of the buffers reclaimed for use when the **bufhigh(***n***)** limit has been reached. A higher value means a reclaim operation is performed less often, but the cached data is significantly trimmed on each reclaim. This can result in poor read performance because readahead buffers might be stolen. Lower values result in more frequent reclaim operations, but the cached data normal water mark is higher, meaning possibly better performance by reading out of cached data. *n* is an integer from 1 to 99 (the default is **20**).

Unsupported Attributes: The following NFS Server attributes are not supported by DFS and are ignored.

- **attrtimeout**
- **noattrtimeout**
- **noreadtimeout**
- **nowritetimeout**
- **readaheadmax**
- **readtimeout**
- **retrieve**
- **retrieve(wait)**
- **writetimeout**.

Examples

The following is an example of an attributes file.

```
startfileattributes:
file: b.c, b.c.d
binary, space(50,0), dsorg(ps)
file: x.y
text
endfileattributes:
blksize(6160)
dir(250)
dsntype(pds)
dsorg(ps)
keys(64,0)
lrecl(80)
recfm(fb)
recordsize(512,4K)
nonspanned
space(100,10), blks
blankstrip
lf
executebiton
nofastfilesize
filetimeout(20)
mapleaddot
maplower
noretrieve
setownerroot
bufhigh(2M)
cachewindow(16)
logicalcache(1M)
maxrdforszleft(32)
percentsteal(20)
```

Note: An example attributes file can be found in **/opt/dfsglobal/examples**.

Implementation Specifics

The attributes file is in either an HFS file, a fixed-block partitioned data set, or a fixed-block sequential data set with a record length of 80. The file's location is specified by the **_IOE_RFS_ATTRIBUTES_FILE** environment variable for the **dfskern** process. For example, **`_IOE_RFS_ATTRIBUTES_FILE=/opt/dfslocal/var/dfs/rfstab`**, this is the default location.

If no attributes are specified for specific files, DFS can use the same attributes file as the DFSMS/MVS NFS Server. For example, **`_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'`**, would cause DFS to use the member NFSSATT in PDS NFSADMIN.NFSS for the DFS record file attributes. (The NFS Server attributes file is specified in the NFSATTR DD statement of the NFS Server startup procedure.) NFS Server attributes that are not supported by DFS are ignored. See "Unsupported Attributes" on page 347 for a list of attributes that are not supported.

A DFS attributes file can also be specified on an RFS fileset basis. It can be specified in the **devtab** entry for the RFS fileset on the same line as the data set name: **attrfile** *attributes_file* (where *attributes_file* is the name of the attributes file that controls the data set creation, processing and site attributes for this RFS fileset). For example,

```
* RFS devices
define_ufs 3 rfs
USERA attrfile /opt/dfslocal/var/dfs/rfstab2
```

If no **attrfile** parameter is specified in the **devtab** entry for the RFS fileset, the attributes are taken from the attributes file specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable in the **dfskern** process.

If this is not specified by the **_IOE_RFS_ATTRIBUTES_FILE** environment variable and if **/opt/dfslocal/var/dfs/rfstab** does not exist then the DFS Server system defaults are used for RFS attributes.

Within a DFS attributes file, file attributes (creation and processing) can be specified on an individual file basis. The attributes file is first scanned for specific file attributes. This is indicated at the beginning of the attributes file by the **startfileattributes:** and **endfileattributes:** keyword pair. Within these two keywords, there may be multiple **file:** keywords. Each **file:** keyword is followed by one or more filenames. The filename specified is the OS/390 data set name minus the exported prefix. The attributes for this file or files are specified on subsequent lines and delimited by another **file:** keyword or the **endfileattributes:** keyword. If a file being accessed is not found in the specific file attributes section of the attributes file, the attributes are taken from the general section of the attributes file.

Related Information

Files:

devtab	**dfsexport**

admin.bak

Purpose

Contains the administrative list for the Backup server.

Usage

The **admin.bak** file is an administrative list of all users and groups who can issue commands in the **bak** command suite. Most commands in the **bak** command suite are used to communicate with the Backup server. The commands in the **bak** suite are used to modify information in the Backup Database and to dump and restore data, as necessary.

A master copy of the backup database resides on one server machine; other server machines (optimally two) house replicated copies of the database. Any machine that houses a copy of the backup database is referred to as a Backup Database machine. The Backup server, or **bakserver** process, must run on all Backup Database machines.

An **admin.bak** file must reside on each Backup Database machine. For the most part, the **admin.bak** file contains the UUIDs of users and groups. However, it must also contain the abbreviated DFS server principals of all Backup Database machines in the local cell to allow the synchronization site for the Backup Database to distribute changes to the secondary sites. The server principals can be present as members of a group included in the list.

Each time the backup server is started on any machine, it automatically creates the **/opt/dcelocal/var/dfs/admin.bak** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command.

Once the **admin.bak** file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.bak** file should be stored in the directory named **/opt/dcelocal/var/dfs** on each Backup Database machine. If it is stored in a different directory, the full pathname of the file must be specified when the backup server is started. Do not create multiple copies of the **admin.bak** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the backup server.

A single version of the **admin.bak** file should be created and maintained on a System Control machine. The **upclient** processes running on the cell's Backup Database machines can then update their local copies of the file using the **upserver** process running on the System Control machine.

Independent versions of the **admin.bak** file should not be maintained on each Backup Database machine in a cell. Because the Backup Database is a **Ubik** database, any of the secondary sites may be obligated to assume the role of synchronization site for the Backup Database at any time. A system administrator listed in the **admin.bak** file on the machine housing the former synchronization site may not be listed in the **admin.bak** file on the machine housing the new synchronization site; the administrator, who could issue commands that affect the backup database on the former machine, may not be able to issue commands that affect the database on the new machine.

Related Information

Commands:

bakserver	**bos lsadmin**	**bos rmadmin**
bos addadmin		

File:
admin.bak

admin.bos

Purpose

Contains the administrative list for the Basic OverSeer (BOS) server.

Usage

The **admin.bos** file is an administrative list of all users and groups that can use the Basic OverSeer server (BOS Server) to manage server processes on a server machine. The **admin.bos** file usually includes the UUIDs of users and groups only; it is not necessary to add a server machine to the **admin.bos** file.

The BOS Server, or **bosserver** process, runs on every DFS server machine in a domain. An **admin.bos** file must reside on each machine running the **bosserver** process.

A user must be represented in the **admin.bos** file on a machine (either directly or indirectly, through a group) to issue commands that affect the server processes on that machine (for example, to create, start, or stop processes). Because system administrators listed in the **admin.bos** file can issue **bos** commands, they can cause DFS server processes to run with DFS authorization checking disabled. Furthermore, because inclusion in the **admin.bos** file gives an administrator such additional privileges, the administrators listed in the **admin.bos** file are usually a subset of the users in the administrative lists for a server machine or domain.

Each time the BOS Server is started on any machine, it automatically creates the **/opt/dcelocal/var/dfs/admin.bos** file if the file does not already exist. Once the file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.bos** file should be stored in the directory named **/opt/dcelocal/var/dfs** on each server machine. If it is stored in a different directory, the full pathname of the file must be specified when the BOS Server is started. Do not create multiple copies of the **admin.bos** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the BOS Server.

It is recommended that a single version of the **admin.bos** file be created and maintained on a domain System Control machine. The **upclient** processes running on the domain's server machines can then reference the file using the **upserver** process running on the System Control machine.

Independent versions of the **admin.bos** file should not be maintained on each server machine in a domain. Doing so may result in a system administrator being permitted to manage processes on one machine but not on another.

Related Information

Commands:

bos addadmin	**bos rmadmin**	**bosserver**
bos lsadmin		

admin.fl

Purpose

Contains the administrative list for the Fileset Location Server.

Usage

The **admin.fl** file is an administrative list of all users and groups who can use the Fileset Location Server (FL Server) to modify the Fileset Location Database (FLDB). A master copy of the FLDB resides on one server machine; other server machines (usually two) house replicated copies of the database. Any machine that houses a copy of the FLDB is referred to as a Fileset Database machine. The FL Server, or **flserver** process, must run on all Fileset Database machines.

An **admin.fl** file must reside on each Fileset Database machine. The **admin.fl** file typically contains the UUIDs of users and groups. However, it must also contain the abbreviated DFS server principals of all Fileset Database machines in the local cell to allow the synchronization site for the FLDB to distribute changes to the secondary sites. The server principals can be present as members of a group included in the list.

Each time the Fileset Location Server is started on any machine, it automatically creates the **/opt/dcelocal/var/dfs/admin.fl** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command.

Once the **admin.fl** file exists, principals and groups can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals and groups currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.fl** file should be stored in the directory named **/opt/dcelocal/var/dfs** on each Fileset Database machine. If it is stored in a different directory, the full pathname of the file must be specified when the FL Server is started. Do not create multiple copies of the **admin.fl** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the FLDB.

It is recommended that a single version of the **admin.fl** file be created and maintained on a System Control machine. The **upclient** processes running on the cell's Fileset Database machines can then reference the file using the **upserver** process running on the System Control machine.

Independent versions of the **admin.fl** file should not be maintained on each Fileset Database machine in a domain. Because the FLDB is a Ubik database, any of the secondary sites may be obligated to assume the role of synchronization site for the FLDB at any time. A system administrator listed in the **admin.fl** file on the machine housing the former synchronization site may not be listed in the **admin.fl** file on the machine housing the new synchronization site. The administrator, who could issue commands that affect the FLDB on the former machine, may not be able to issue commands that affect the database on the new machine, or vice versa.

Related Information

Commands:

bos addadmin	**bos rmadmin**	**flserver**
bos lsadmin		

admin.ft

Purpose

Contains the administrative list for the Fileset Server.

Usage

The **admin.ft** file is an administrative list of all principals and groups that can use the Fileset Server to manipulate filesets on a File Server machine. The **admin.ft** file includes the UUIDs of users and groups who can issue commands that affect a machine's filesets; it includes the UUIDs of servers from which the machine can accept filesets.

A File Server machine is defined as any machine that exports data for use in the global namespace. The Fileset Server, or **ftserver** process, runs on every File Server machine in a domain. The **ftserver** process provides the interface for any commands that affect filesets on a File Server machine. An **admin.ft** file must reside on each machine running the **ftserver** process.

A user must be represented in the **admin.ft** file on a machine (either directly or indirectly, through a group) to issue commands that affect the filesets on a machine (for example, to create, move, delete, back up, or restore a fileset). The user must also be listed in the file to move filesets onto the machine from a different machine. In addition, the principal name for a server machine must be included in the **admin.ft** file on another machine if filesets are to be moved from it to the other machine.

Each time the Fileset Server is started on any machine, it automatically creates the **/opt/dcelocal/var/dfs/admin.ft** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command.

Once the **admin.ft** file exists, principals can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.ft** file should be stored in the directory named **/opt/dcelocal/var/dfs** on each File Server machine. If it is stored in a different directory, the full pathname of the file must be specified when the Fileset Server is started. Do not create multiple copies of the **admin.ft** file and store them in different directories on the same machine; unauthorized users may be able to use the extraneous copies to access the Fileset Server or to allow the File Server machine to accept filesets from unprivileged machines.

It is recommended that a single version of the **admin.ft** file be created and maintained on a domain's System Control machine. The **upclient** processes running on the domain's file server machines can then reference the file using the **upserver** process running on the System Control machine.

Independent versions of the **admin.ft** file should not be maintained on each File Server machine in a domain. Doing so may result in a system administrator being permitted to manipulate filesets on one machine but not on another, or it may result in the administrator being able to move filesets among only some of the machines in the domain.

Related Information

Commands:

bos addadmin	**bos rmadmin**	**ftserver**
bos lsadmin		

admin.up

Purpose

Contains the administrative list for the Update Server.

Usage

The **admin.up** file is an administrative list of all server principals that can receive copies of files using the Update Server. The **admin.up** file usually contains the UUIDs of server machines only; it is not necessary to add users or groups to the **admin.up** file.

The Update Server distributes files such as common configuration files, binary files, and administrative lists from System Control and Binary Distribution machines to the other server machines in a domain. Server machines that rely on System Control and Binary Distribution machines for these kinds of files run the **upclient** process, the client portion of the Update Server. System Control and Binary Distribution machines run the **upserver** process, the server portion of the Update Server.

Important Note to Users

OS/390 DFS supports only the System Control machine functions. Binary Distribution machine functions are not supported.

Each instance of the **upclient** process frequently checks with the **upserver** process on the System Control and Binary Distribution machines to ensure that its local copies of the proper files are current. If newer versions of the files exist, the **upclient** process retrieves them from the **upserver** process and installs them in place of the outdated versions of the files. The **admin.up** file resides on machines running the **upserver** process; it specifies the machines whose **upclient** processes are permitted to obtain copies of files from the **upserver** process.

Each time the **upserver** process is started on any machine, it automatically creates the **/opt/dcelocal/var/dfs/admin.up** file if the file does not already exist. You can also create the file by including the **-createlist** option with the **bos addadmin** command.

Once the **admin.up** file exists, principals can be added to it with the **bos addadmin** command, and they can be removed from it with the **bos rmadmin** command. The **bos lsadmin** command can be used to list the principals currently in the file. Because administrative lists are stored as binary files, you must use these commands to modify them; you cannot edit them directly.

The **admin.up** file should be stored in the directory named **/opt/dcelocal/var/dfs** on each machine running the **upserver** portion of the Update Server. If it is stored in a different directory, the full pathname of the file must be specified when the **upserver** process is started. Do not create multiple copies of the **admin.up** file and store them in different directories; unauthorized users may be able to use the extraneous copies to have the **upserver** process allow unprivileged machines to obtain copies of files.

Implementation Specifics

OS/390 DFS supports only the System Control machine functions. Binary Distribution machine functions are not supported.

Related Information

Commands:

bos addadmin	**bos rmadmin**	**upserver**
bos lsadmin	**upclient**	

BakLog

Purpose

Contains messages generated by the Backup Server.

Usage

The **BakLog** file contains execution and error messages generated by the Backup Server (**bakserver** process). The Backup Server runs on every Backup Database machine in a cell, providing the interface by which authorized users can modify the Backup Database.

If the **BakLog** file does not already exist when the Backup Server starts, the server process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Backup Server starts, the process moves the current version of the file to the **BakLog.old** file in the same directory (overwriting the current **BakLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. This file is stored in EBCDIC on OS/390. It can also be viewed in the same manner as other OS/390 HFS files.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help in evaluating server process failures and other problems.

Implementation Specifics

This file is stored in EBCDIC on OS/390. It can be viewed in the same manner as other OS/390 HFS files.

Related Information

Commands:

bakserver	**bos getlog**

BosConfig

Purpose

Defines server processes to be monitored by the Basic OverSeer (BOS) Server.

Usage

The **BosConfig** file defines the server processes to be monitored by the BOS Server (**bosserver** process) on a server machine. It contains a process entry for each process to be monitored by the BOS Server; each entry defines how its process is to run. The **BosConfig** file also maintains both the weekly and daily restart times for the BOS Server and processes that have entries in the file.

The BOS Server runs on each server machine, continually monitoring and, if necessary, restarting the other server processes on the machine. The BOS Server checks the **BosConfig** file whenever it starts or restarts; the information is then transferred into memory and the file is not read again until the BOS Server restarts. Thus, server processes can be started or stopped, independently of their process entries, based on their status in the BOS Server's memory. The order in which process entries appear in the **BosConfig** file is irrelevant.

The **BosConfig** file must reside in the directory named **/opt/dcelocal/var/dfs** on the local disk of a server machine running the BOS Server. The BOS Server creates a **BosConfig** file with only default restart times and no process entries if the file does not exist when the BOS Server starts. Because it is a local file, the information it contains can be different for different machines.

Each process entry in a **BosConfig** file includes the following information about the process:

Name
This is the name used by the BOS Server to refer to the process. Although any name can be chosen, the following names are recommended for consistency:

ftserver	For the Fileset Server process
flserver	For the Fileset Location Server process
upclient	For the client portion of the Update Server
upserver	For the server portion of the Update Server
repserver	For the Replication Server process
bakserver	For the Backup Server process

Type
A process can be one of two types:

simple	A continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes.
cron	A process that runs independently of any other processes; however, unlike a **simple** process, a **cron** process runs periodically, not continuously.

Status flag
Status flags are for internal use only; they do not appear in any output. A process can have one of two status flags:

Run	Means the process is to run whenever possible; the BOS Server starts it automatically at re-boot and restarts it automatically if it fails. (The **Run** status flag appears in the file as a **1**.)
NotRun	Means the BOS Server does not start or restart the process. (The **NotRun** status flag appears in the file as a **0**.)

Command parameters

The BOS Server uses these parameters to run the process. For a **simple** process, a single command parameter specifying the complete pathname of the binary file for a DFS command or any other command to be run is used.

Important Note to Users

On OS/390 DFS, **simple** processes are not identified by complete pathnames to the binary files for the processes. All **simple** process commands in OS/390 DFS are members of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) created during installation (for further information, refer to the *OS/390 Program Directory*). As all OS/390 member names are limited to eight or less characters, the **repserver** and **bakserver** process member names are shortened to **rpserver** and **bkserver**, respectively. For additional information, see the examples on page 488.

The following are the PDS member names for the DFS processes:

bkserver Starts the Backup Server process. The name, **bkserver**, is an alias for the load library entry, **IOEBKSRV**. For further information, see “bakserver” on page 473.

flserver Starts the Fileset Location Server process. The name, **flserver**, is an alias for the load library entry, **IOEFLSRV**. For further information, see “flserver” on page 605.

ftserver Starts the Fileset Server process. The name, **ftserver**, is an alias for the load library entry, **IOEFTSRV**. For further information, see “ftserver” on page 720.

upclient Starts the Update Client process. The name, **upclient**, is an alias for the load library entry, **IOEUPCLN**. For further information, see “upclient” on page 765.

upserver Starts the Update Server process. The name, **upserver**, is an alias for the load library entry, **IOEUPSRV**. For further information, see “upserver” on page 768.

rpserver Starts the Replication Server process. The name, **rpserver**, is an alias for the load library entry, **IOERPSRV**. For further information, see “repserver” on page 745.

For a **cron** process, two command parameters are used: the complete pathname of the binary file for a DFS command or any other commands to be executed, and the time the BOS Server is to execute the command.

Note: For a **cron** process on OS/390 DFS, two command parameters are used: the partitioned data set (PDS) member name of the DFS process to be executed, and the time the BOS Server is to execute the command.

Although it is an EBCDIC file, do not edit the **BosConfig** file directly; always use the appropriate **bos** commands. Editing the file directly can introduce changes the BOS Server does not recognize until it is restarted and again reads the file.

The following **bos** commands modify process entries or restart times in the **BosConfig** file:

bos create
Adds a process entry to the file, setting the process' status to **Run** in both the file and memory, and starts the process

bos delete
Removes a process entry for a stopped process from the file

bos stop Stops a running process by changing its status to **NotRun** in both the file and memory

bos start Starts a stopped process by changing its status to **Run** in both the file and memory

bos setrestart
Sets the weekly and daily restart times included in the file

The following **bos** commands access process entries in the **BosConfig** file:

bos status
Lists the statuses of server processes on a machine, from which you can determine information about their process entries

bos restart
Stops and immediately restarts processes that have process entries in the file

bos getrestart
Displays both the weekly and daily restart times from the file

Additional **bos** commands can be used to start or stop a process by changing its status in the BOS Server's memory without affecting its process entry in the **BosConfig** file.

Cautions

Do not edit the **BosConfig** file directly. Always use the appropriate **bos** commands to manipulate process entries in the **BosConfig** file. Editing the file directly can introduce changes that the BOS Server is not aware of until it is restarted and again reads the file.

Examples

The following **bos create** command creates a **cron** process entry named **backup** in the **BosConfig** file on the machine named **aixfs1**. It specifies that a **cron** process identified by **backup** is to use the **fts clonesys** command daily at 5:30 a.m. to create backup versions of all read-write filesets on **aixfs1.abc.com**. The **-localauth** option is used with the **fts clonesys** command to use the identity of the local machine as the identity of the issuer of the command.

```
$ bos create /.../abc.com/hosts/aixfs1 backup cron \
"/opt/dfsglobal/bin/fts clonesys -s /.../abc.com/hosts/aixfs1 -localauth" 5:30
```

For OS/390 DFS, the pathname for the program to be executed is not needed, instead the PDS member name is used. In the following example, the **fts clonesys** command is run at 5:30 am.

```
$ bos create /.:/hosts/mvsfs1 backup cron \
"IOEFTS clonesys -s /.:/hosts/mvsfs1 -localauth" 5:30
```

The following **bos setrestart** command sets the general restart time when the BOS Server restarts itself and all of the processes with entries in the **BosConfig** file. It specifies that all processes, including the **bosserver** process, on **fs1.abc.com** are to be restarted every Sunday morning at 4:00 a.m.

```
$ bos setrestart /.../abc.com/hosts/fs1 -general "sun 4:00"
```

Implementation Specifics

The **BosConfig** file is stored as an EBCDIC file on OS/390 DFS.

Two command parameters are used with a **cron** process on OS/390 DFS; the partitioned data set (PDS) member name of the DFS process to be executed, and the time the BOS Server is to execute the command.

Related Information

Commands:

bos create	**bos setrestart**	**bos stop**
bos delete	**bos start**	**bosserver**

BosLog

Purpose

Contains messages generated by the Basic OverSeer (BOS) Server.

Usage

The **BosLog** file contains execution and error messages generated by the Basic OverSeer (BOS) Server (**bosserver** process). The BOS Server runs on every server machine in a cell, monitoring the other server processes on the machine and restarting them as necessary.

If the **BosLog** file does not already exist when the BOS Server starts, the server process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the BOS Server starts, the process moves the current version of the file to the **BosLog.old** file in the same directory (overwriting the current **BosLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. This file is stored in EBCDIC on OS/390. It can also be viewed in the same manner as other OS/390 HFS files.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help you evaluate server process failures and other problems.

Implementation Specifics

The **BosLog** file is stored as an EBCDIC file in OS/390 DFS.

Related Information

Commands:

bos getlog	**bosserver**

CacheInfo

Purpose

Defines the initial configuration of the Cache Manager.

Usage

The **CacheInfo** file specifies the initial configuration of the Cache Manager on a client machine. The Cache Manager checks the file at initialization to determine certain cache configuration information. It uses the file regardless of the type of caching, disk or memory, in use on the machine.

The **CacheInfo** file is created during DFS client installation. It must reside in the directory named **/opt/dcelocal/etc**.

The file is a one-line EBCDIC file consisting of the following three fields separated by colons:

- The first field names a directory on the local disk where the Cache Manager mounts the DCE global namespace. The default entry is the global namespace designation (**/...**). If **/...** is not specified, symbolic links to the global namespace fail.

  The value of this field can be overridden with the **-mountdir** option of the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file.

- The second field specifies the minor device number for a local DCE Local File System aggregate that serves as the DFS cache for a disk cache. This is the aggregate in which the Cache Manager stores the **V***n*, **CacheItems**, and **FilesetItems** files that it creates. There is no default for this entry. The entry must be specified. Although the indicated aggregate is not used with a memory cache, an entry must appear in this field even if memory caching is employed on the machine.

  The value of this field can be overridden with the **-cachedir** option of the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file.

- The third field specifies the cache size in 1024-byte (1-kilobyte) blocks. The amount of disk space or machine memory used for caching depends on several factors. The size of the DCE Local File System aggregate that houses the cache or the amount of memory available on the machine places a limit on the cache size. Do not use more than 75% of the DCE Local File System aggregate for a disk cache.

  The value of this field can be overridden with the **-blocks** option of the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file. The disk cache size can also be overridden with the **cm setcachesize** command. The **cm getcachesize** command can be used to view the current size of the cache (disk or memory) and the amount in use.

Because it is an EBCDIC file, the **CacheInfo** file can be directly modified with a text editor. To modify the file, log in as **root** on the machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Cautions

The size of the DCE Local File System aggregate that houses the cache or the amount of memory available on the machine places a limit on the cache size. Do not use more than 75% of the DCE Local File System aggregate for a disk cache.

Be precise when editing the **CacheInfo** file; use colons to separate the fields in the file, and do not include any spaces in the file.

Examples

An example of a typical **CacheInfo** file follows. It lists the DCE global namespace mounted at the global namespace designation (**/...**), the minor device number of 998 for the local DCE Local File System aggregate used for the cached file, and a defined cache size of 50,000 1-kilobyte blocks.

```
/...:998:50000
```

Implementation Specifics

The **CacheInfo** file is stored as an EBCDIC file on OS/390 DFS.

Related Information

Commands:		
cm getcachesize	**cm setcachesize**	**dfsd**
Files:		
CacheItems	**FilesetItems**	**Vn**

CacheItems

Purpose

Records information about each V file in a disk cache.

Usage

The **CacheItems** file is a binary file created and maintained by the Cache Manager for its own use and for use by developers for debugging. It records information about each V file on a client machine using a disk cache. The information includes the file ID number and data version number of each V file.

The **CacheItems** file always resides in the DCE Local File System aggregate where its minor device number is specified in the second field of the **CacheInfo** file; it can be overridden to name a different minor device number.

Implementation Specifics

In OS/390 DFS, you cannot access the **CacheItems** file through OS/390 UNIX. The DCE Local File System aggregate where this file resides is not locally mounted. Always use the commands provided with DFS to alter this file. For further information, see Chapter 13, “Configuring the Cache Manager” on page 251.

Related Information

Files:

CacheInfo	**Vn**

cm attributes file (cmattr)

Purpose

Contains directives that map a filename extension (suffix) to an indication whether the OS/390 DFS Cache Manager should translate the file data from ASCII to EBCDIC and vice versa (encoding).

Usage

The cmattr file is a text file that is stored in HFS. The Cache Manager locates the cmattr file during start-up (or restart) by examining the _IOE_CM_ATTRIBUTES_FILE environment variable for the DFSCM process. The cmattr file has the following format:

AddType *.suffix representation encoding* **[***quality***]**

The Cache Manager only examines the first, second and fourth fields. The others are ignored. Comments can be created by using the # character. All fields are case sensitive.

AddType A keyword that indicates a directive to map a suffix to an encoding.

.suffix The file name suffix. Wildcard characters are not allowed.

representation
The MIME type and subtype you want to bind to files that match the corresponding suffix. This field is ignored by DFS.

encoding The type of data the file contains. The only value that the Cache Manager looks for is ebcdic. This means that incoming data should be translated from ASCII (ISO 8859-1) to EBCDIC (IBM-1047 or the current codepage for the DFSCM process). Outgoing data should be translated from EBCDIC to ASCII. All other values for encoding are ignored and the data is not translated. Before data is translated, it is checked for valid characters to avoid translating data that is already in the correct format.

quality This is an optional indicator of the relative (on a scale of 0.0 to 1.0) for the content type. This field is ignored by DFS.

If the file name suffix is not found in the **cmattr** file or the file name has no suffix, then translation is determined by the **-translation** DFSCM initialization parameter. See "dfsd" on page 589.

The **cmattr** AddType directive has the same format as the WebSphere Application Server uses in its configuration file (httpd.conf). You can point the Cache Manager to this file. All other directives are ignored.

Examples

The following is an example of a **cmattr** file:

```
# Map suffixes to the encoding
AddType  .bin  application/octet-stream  binary  1.0
AddType  .ps   application/postscript    ebcdic  0.8 # PostScript
AddType  .PS   application/postscript    ebcdic  0.8 # PostScript
AddType  .c    text/plain                ebcdic  0.5 # C source
AddType  .html text/html                 ebcdic  1.0 # HTML
AddType  .htm  text/html                 ebcdic  1.0 # HTML on PCs
AddType  .gif  image/gif                 binary  1.0 # GIF
```

CMLog

Purpose

Contains messages generated by the Cache Manager.

Usage

The **CMLog** file contains execution and error messages generated by the Cache Manager (**DFSCM**). The Cache Manager runs on every client machine in a cell, handling file requests for data exported by DFS File Servers.

If the **CMLog** file does not already exist when the Cache Manager is started, the client process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Cache Manager starts, the Cache Manager moves the current version of the file to the **CMLog.old** file in the directory (overwriting the current **CMLog.old** file if it exists) before creating a new version to which to append messages.

The Cache Manager can write different types of output to the file, depending on the actions it performs and any problems it encounters. This file is stored in EBCDIC on OS/390. It can be viewed in the same manner as other OS/390 HFS files. The contents of the log file can help you evaluate Cache Manager failures and other problems.

Implementation Specifics

The **CMLog** file is stored as an EBCDIC file in OS/390 DFS. The logging of these messages is controlled by the **-log {on | off}** option documented in the **dfsd** command. **dfsd** options are specified in the **_IOE_CM_PARMS** environment variable contained in the **/opt/dfslocal/home/dfscm/envar** file.

devtab

Purpose

Stores identifying information for all DCE Local File System logical volumes and non-Local File System devices to be exported.

Format

The **devtab** file contains the following lines:

***** *comment*
define_lfs *n* **|** **define_ufs** *n* **[<u>hfs</u> | rfs]**
{*linear-data-set-name* **|**
hfs-data-set-name **[text | binary | auto] |**
rfs-data-set-prefix **[text | binary] [attrfile** *attributes-file***] |**
rfs-data-set-name **[text | binary] [attrfile** *attributes-file***]}**

Options

***** *comment*
Specifies a comment line.

define_lfs *n*
Defines a DCE Local File System aggregate logical volume, *n* specifies a minor device number which is a unique identifier that can be any number greater than zero.

define_ufs *n* **[<u>hfs</u> | rfs]**
Defines a non-Local File System aggregate, *n* specifies a minor device number which is a unique identifier that can be any number greater than zero. Specifying **hfs** (Hierarchical File System) or **rfs** (Record File System) indicates the particular non-Local File System. The default is **hfs**.

linear-data-set-name
Specifies the name or names of the linear data sets that make up the DCE Local File System logical volume.

hfs-data-set-name
Identifies the data set name of the HFS file system that you want exported.

rfs-data-set-prefix
Identifies the prefix of the record data sets that you want exported.

rfs-data-set-name
Identifies the data set name of a Partitioned Data Set (PDS) or Partitioned Data Set Extended (PDSE) that you want exported.

attrfile *attributes-file*
Identifies the name of the attributes file to be used for this RFS fileset.

text | binary
Specifies whether the data needs to be translated (**text**) or not (**binary**). The default is controlled by the **_IOE_HFS_TRANSLATION** environment variable setting in the **dfskern** process (for HFS filesets) and by the **_IOE_RFS_TRANSLATION** environment variable setting in the **dfskern** process (for RFS filesets).

auto
Specifies that the decision to translate HFS data is based on whether the first 255 bytes of the file are deemed to be valid characters.

Usage

The **devtab** file is used to define the logical volumes for DCE Local File System aggregates and the linear data sets that make up each DCE Local File System aggregate to be exported. The file is also used to define non-Local File Systems you wish to export. The **devtab** file resides in the directory named **/opt/dcelocal/var/dfs**.

The **devtab** file is an EBCDIC file that can be edited with a text editor. You must have write and execute permissions on the **/opt/dcelocal/var/dfs** directory to create the file. You must have write permission on the file to edit it.

To create a DCE Local File System aggregate, a logical volume must be defined. The linear data sets that make up the logical volume for each aggregate you choose to export must also be specified. A DCE Local File System logical volume is defined by entering **define_lfs** *n* in the **/opt/dcelocal/var/dfs/devtab** file. A minor device number, *n*, is assigned to each DCE Local File System aggregate. The minor device number is a unique identifier that can be any number greater than zero. This number becomes part of the name of the defined logical volume. The linear data sets that make up the logical volume are defined after the logical volume definition.

A non-Local File System aggregate is created by defining the type of file system to be exported in the **devtab** file. Entering **define_ufs** *n* to the **devtab** file, defines the type of file system as **ufs**, a non-Local File System, and maps a unique minor device number, *n*, to the non-Local File System you wish to export. You can also enter **define_ufs** *n* **rfs** to the devtab file, where **rfs** maps a minor device number to the RFS fileset you want to export. The minor device number is a unique identifier that can be any number greater than zero. This number becomes part of the name of the device name. Non-Local File System aggregates are a single complete fileset in DFS. Each non-Local File System to be exported must have a unique logical volume defined. The name of the non-Local File System that comprises the aggregate is entered in the **devtab** file after the logical volume definition. You can also specify an optional character data translation parameter on the same line after the name of the non-Local File System. The possible values for the translation control parameter are the following:

binary
: Do not translate the data.

text
: Translate the data with the default translation tables. The default for local data is the local code page for the process. The code page for "wire" data is ISO 8859-1. "Examples" on page 370 shows an example using the **text** parameter.

An additional translation control parameter value for HFS is available for HFS filesets and an additional translation configuration file is available for HFS.

The additional translation control parameter value for HFS is:

auto
: Determine whether to translate the data based on the contents of the data. The algorithm is as follows:

- for outgoing data (read), if the first 255 bytes of data are valid EBCDIC characters then translate to ASCII
- for incoming data (write), if the first 255 bytes of data are valid ASCII characters then translate to EBCDIC.

If the translation control parameter is omitted the default is controlled by the **_IOE_HFS_TRANSLATION** environment variable setting in the **dfskern** process (for HFS filesets) and in the **_IOE_RFS_TRANSLATION** environment variable setting in the **dfskern** process (for RFS filesets), see Appendix B, "Sharing Data Between OS/390 UNIX Applications, Commands, and DFS Clients" on page 795 for more information.

If the non-Local File System is an RFS fileset, you can also optionally specify the name of an attributes file on the same line after the *rfs-data-set-prefix* or *rfs-data-set-name* by using the **attrfile** keyword. See "Attributes File (rfstab)" on page 341 for more information on the attributes file.

The minor device number, *n*, need not be a unique number across both DCE Local File System and non-Local File System aggregates.

Note: On DFS, a non-Local File System file is either an OS/390 HFS file system or a record data set (or a set of record data sets with a common prefix) that DFS calls a Record File System (RFS). An OS/390 HFS aggregate is a single OS/390 HFS file and is a non-Local File System aggregate. The aggregate is a complete fileset to the DFS server. Before exporting an OS/390 HFS fileset, the OS/390 HFS file **must** be mounted to the OS/390 address space. For information about allocating and mounting OS/390 HFS file systems, see *OS/390 UNIX System Services Planning* book, SC28-1890.

Examples

The following examples show **devtab** entries for DCE Local File System and non-Local File System aggregates. All entries beginning with an asterisk (*) are comment lines.

In the first example, a DCE Local File System device name is defined by entering **define_lfs 1** in the **devtab** file. A minor device number of **1** is assigned. The lines following the definition of DCE Local File System logical volume specify the names of the linear data sets that make up the DCE Local File System logical volume. The DCE Local File System device name is **/dev/lfs1** with **1** representing the device's minor number.

```
* Devtab - Example Entry for a logical volume
define_lfs 1
DFS.DCELFS.AGGR001.LDS00001
DFS.DCELFS.AGGR001.LDS00002
DFS.DCELFS.AGGR001.LDS00003
```

The second example shows a **devtab** entry for a non-Local File System HFS device. The first line, **define_ufs 2**, defines the type of file system as **ufs**, a non-Local File System. A unique minor device number of **2** is assigned. The line following the definition of the non-Local File System device, **omvs.user.abc**, identifies the name of the OS/390 HFS file system to be exported and specifies a translation control parameter of **text**. A translation control parameter of **text** means data is translated. The non-Local File System device name is **/dev/ufs2** with **2** representing the device's minor number.

```
* HFS devices
define_ufs 2
omvs.user.abc text
```

The third example shows a **devtab** entry for a non-Local File System RFS device. The first line, **define_ufs 3 rfs** defines the type of non-Local File System (**rfs**) and a minor device number of **3**. The following line, **USERA** is the prefix of the record data sets that you want to export as a single RFS fileset. The non-Local File System device name is **/dev/ufs3** with **3** being the device's minor number.

```
* RFS devices
define_ufs 3 rfs
USERA
```

Note: An example **devtab** file can be found in **/opt/dfsglobal/examples**.

In summary, the **define_lfs 1** entry in the **devtab** corresponds to the **/dev/lfs1** entry in the **dfstab**, the **define_ufs 2** entry in the **devtab** corresponds to the **/dev/ufs2** entry in the **dfstab**, and the **define_ufs 3 rfs** entry in the **devtab** corresponds to the **/dev/ufs3** entry in the **dfstab**.

Implementation Specifics

The **devtab** file is stored as an EBCDIC file in OS/390 DFS.

Related Information

Commands:	
dfsexport	**bosserver**
File:	
Attributes	**dfstab**

dfstab

Purpose

Specifies DCE Local File System aggregates and non-Local File System partitions that can be exported.

Usage

The **dfstab** file includes information about each DCE Local File System aggregate and each non-Local File System partition that can be exported from the local disk to the DCE namespace. The file is read by the **dfsexport** command, which exports specified aggregates and partitions to the DCE namespace. (It is also read by the **newaggr** command, which initializes DCE Local File System aggregates, the **growaggr** command, which is used to increase the size of a DCE Local File System aggregate, and the **salvage** command, which is used to recover, verify, or salvage the structure of a DCE Local File System.) The **dfstab** file must reside in the directory named **/opt/dcelocal/var/dfs**. The **dfsexport** command looks in that directory for the file; if the file is not there, no aggregates or partitions can be exported.

The **dfstab** file is an EBCDIC file that can be edited with a text editor. You must have write and execute permissions on the **/opt/dcelocal/var/dfs** directory to create the file. You must have write permission on the file to edit it.

The file contains a one-line entry for each aggregate or partition available for exporting. Each entry in the file must appear on its own line. The fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab. Because DCE Local File System aggregates contain an arbitrary number of filesets, *do not include a fileset ID number when creating an entry for a DCE Local File System aggregate*.

Device name
The block device name of the aggregate or partition to be exported; for example, **/dev/ufs2**.

Aggregate name
The name to be associated with the aggregate or partition being exported. An aggregate name can contain any characters, but it can be no longer than 31 characters. It must be different from any other aggregate name in the **dfstab** file. Aggregate names cannot be abbreviated, so you should choose a short, descriptive name; for example, **lfs1**.

File system type
The identifier for the type of file system housing the aggregate or partition. For DCE Local File System aggregates, this must be **lfs**; for non-Local File System partitions, it must be **ufs**. Enter the identifier in all lowercase letters.

Aggregate ID
A positive integer different from any other aggregate ID in the **dfstab** file. In the entry for a non-Local File System partition, this field must contain the aggregate ID number specified with the **-aggrid** option of the **fts crfldbentry** command.

Fileset ID
The unique fileset ID number to be associated with the fileset on a non-Local File System partition; for example, 0,,18756. When creating an entry for a non-Local File System partition, this field must contain the fileset ID number generated with the **fts crfldbentry** command. *Do not include a fileset ID number with an entry for a DCE Local File System aggregate.*

When the **dfsexport** command is executed, it reads the **dfstab** file to verify that each aggregate or partition to be exported is listed in the file. An aggregate or partition must have an entry in the **dfstab** file if it is to be exported. To ensure that it does not export an aggregate or partition that is currently exported, the **dfsexport** command refers to a list of all currently exported aggregates and partitions that exist in the kernel of the local machine.

Cautions

Do not change the aggregate ID number assigned to an aggregate or partition in this file once Fileset Location Database (FLDB) entries have been created for filesets on the aggregate or partition. Changing the aggregate ID number used for an aggregate or partition in this file invalidates existing FLDB entries for filesets on the aggregate or partition.

Example

The following **dfstab** file specifies that one non-Local File System partition (**/dev/ufs2**) and two DCE Local File System aggregates (**/dev/lfs3** and **/dev/lfs4**) can be exported:

```
/dev/ufs2      hfs2  ufs   1   0,,18756
/dev/lfs3      lfs1  lfs   3
/dev/lfs4      lfs2  lfs  11
```

Note: You can put comments in the **dfstab** file. Comments have a **#** in column 1.

The following **dfstab** file specifies that one DCE Local File System aggregate (**/dev/lfs1**) and two non-Local File System partitions (**dev/ufs2** and **dev/ufs3** for RFS) can be exported:

```
/dev/lfs1      lfs1  lfs   1
/dev/ufs2      ufs2  ufs   101   0,,1715
/dev/ufs3      rfs3  ufs   102   0,,1718
```

Note: An example **dfstab** file can be found in **/opt/dfs/examples**.

Implementation Specifics

The **dfstab** file is stored as an EBCDIC file in OS/390 DFS.

Related Information

Commands:

dfsexport	**fts crfldbentry**

File:
devtab

FilesetItems

Purpose

Records location mappings for filesets accessed by a Cache Manager using a disk cache.

Usage

The **FilesetItems** file is a binary file created and maintained by the Cache Manager for its own use and for use by developers for debugging. It stores the fileset-to-mount point mapping for each fileset accessed by a Cache Manager using a disk cache. The mappings enable the Cache Manager to respond correctly to operating system and related commands such as **pwd**.

The **FilesetItems** file always resides in the DCE Local File System aggregate where it's minor device number is specified in the second field of the **CacheInfo** file; it can be overridden to name a different minor device number.

Implementation Specifics

In OS/390 DFS, you cannot access the **FilesetItems** file through OS/390 UNIX. The DCE Local File System aggregate where this file resides is not locally mounted. Always use the commands provided with DFS to alter this file. For further information, see Chapter 13, “Configuring the Cache Manager” on page 251.

Related Information

File:
CacheInfo

FlLog

Purpose

Contains messages generated by the Fileset Location Server.

Usage

The **FlLog** file contains execution messages and error messages generated by the Fileset Location Server (**flserver**) process. The Fileset Location Server runs on every Fileset Database machine in a cell, providing the interface by which authorized users can modify the Fileset Location Database (FLDB).

If the **FlLog** file does not already exist when the Fileset Location Server starts, the server process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Fileset Location Server starts, the process moves the current version of the file to the **FlLog.old** file in the same directory (overwriting the current **FlLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command Because it is an EBCDIC file on OS/390 DFS, it can also be viewed in the same manner as other OS/390 UNIX files which require **read** permission on the file.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help in evaluating server process failures and other problems.

Implementation Specifics

The **FlLog** file is stored as an EBCDIC file on OS/390 DFS.

Related Information

Commands:

bos getlog	**flserver**

FMSLog

Purpose

Lists the output of the **fms** command.

Usage

The **FMSLog** file lists the output generated by the **fms** (file mark size) command. The **fms** command determines the tape capacity and end-of-file (EOF) mark size for a tape drive. The command both displays its output on the screen and writes it to the **FMSLog** file, which it creates in the directory from which it is issued.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **FMSLog** file is not necessary on OS/390 DFS and not available.

FtLog

Purpose

Contains messages generated by the Fileset Server.

Usage

The **FtLog** file contains execution messages and error messages generated by the Fileset Server (**ftserver** process). The Fileset Server runs on every File Server machine in a cell. It provides the interface for any commands that affect filesets on a File Server machine.

If the **FtLog** file does not already exist when the Fileset Server starts, the server process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Fileset Server starts, the process moves the current version of the file to the **FtLog.old** file in the same directory (overwriting the current **FtLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. This file is stored in EBCDIC on OS/390 DFS. It can also be viewed in the same manner as other OS/390 files.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help in evaluating server process failures and other problems.

Implementation Specifics

The **FtLog** file is stored as an EBCDIC file in OS/390 DFS.

Related Information

Commands:

bos getlog	**ftserver**

hfs attributes file (hfsattr)

Purpose

Contains directives that map a filename extension (suffix) to an indication whether the OS/390 DFS File Server (DFSKERN) should translate the file data from ASCII to EBCDIC and vice versa (encoding).

Usage

The **hfsattr** file is a text file that is stored in HFS. The File Server locates the hfsattr file during start-up (or restart) by examining the _IOE_HFS_ATTRIBUTES_FILE environment variable for the DFSKERN process. The hfsattr file has the following format:

AddType *.suffix representation encoding* **[***quality***]**

The File Server only examines the first, second, and fourth fields. The others are ignored. Comments can be created by using the # character. All fields are case sensitive.

AddType A keyword that indicates a directive to map a suffix to an encoding.

.suffix The file name suffix. Wildcard characters are not allowed.

representation
The MIME type and subtype you want to bind to files that match the corresponding suffix. This field is ignored by DFS.

encoding The type of data the file contains. The only value that the File Server looks for is ebcdic. This means that incoming data should be translated from ASCII (ISO 8859-1) to EBCDIC (IBM-1047 or the current codepage for the DFSKERN process). Outgoing data should be translated from EBCDIC to ASCII. All other values for encoding are ignored and the data is not translated. Before data is translated, it is checked for valid characters to avoid translating data that is already in the correct format.

quality This is an optional indicator of the relative (on a scale of 0.0 to 1.0) for the content type. This field is ignored by DFS.

If the file name suffix is not found in the **hfsattr** file or the file name has no suffix, then translation is determined by the DFSKERN **_IOE_HFS_TRANSLATION** environment variable. For more information on the **_IOE_HFS_TRANSLATION** environment variable, see page 781.

The **hfsattr** AddType directive has the same format as the WebSphere Application Server uses in its configuration file (httpd.conf). You can point the File Server to this file. All other directives are ignored.

Note: The **hfsattr** file has exactly the same format as the **cmattr** file. You can use the same file for **hfsattr** that you use for **cmattr**.

Examples

The following is an example of a **hfsattr** file:

```
# Map suffixes to the encoding
AddType  .bin  application/octet-stream  binary  1.0
AddType  .ps   application/postscript    ebcdic  0.8 # PostScript
AddType  .PS   application/postscript    ebcdic  0.8 # PostScript
AddType  .c    text/plain                ebcdic  0.5 # C source
AddType  .html text/html                 ebcdic  1.0 # HTML
AddType  .htm  text/html                 ebcdic  1.0 # HTML on PCs
AddType  .gif  image/gif                 binary  1.0 # GIF
```

Identity Mapping Input File

Purpose

Maps a DCE user ID to an OS/390 user ID. The mapping is used to determine the file access permissions for exported HFS and RFS file systems.

Note: RACF may be used for the Identity Mapping function (see "Using the RACF Identity Mapping Function Method" on page 178 for more information). In this case, the Identity Mapping input file is not needed. The **_IOE_MVS_IDMAP_SAF** environment variable in the **dfskern** process controls whether the Identity Mapping output file or RACF is used for the Identity Mapping function. If the **_IOE_MVS_IDMAP_SAF** environment variable is not specified or is set to OFF the Identity Mapping output file is used. If the **_IOE_MVS_IDMAP_SAF** environment variable is set to ON RACF is used.

Usage

The Identity Mapping input file is a text file that the administrator creates and maintains. This must be created as an HFS file. The file is specified as the first parameter of the **mapid** command. For further information, see "mapid" on page 735.

Note: The DFS server administrator must also map their DCE user ID to their OS/390 user ID.

The Identity Mapping input file contains one or more identity mapping declarations and has the following general format:

DCE-user-ID1
OS/390-user-ID1
DFS-server-name

DCE-user-ID2
OS/390-user-ID2
...

This format illustrates the two types of entries that can exist on the Identity Mapping input file. The first entry has three elements: DCE user ID, OS/390 user ID, and DFS server name. The second entry only has two elements: DCE user ID and OS/390 user ID. A blank line is required between entries. The following list explains each element in an identity mapping entry:

DCE-user-ID
Is the client's DCE identity. This may either be a simple DCE principal name (for clients within the cell) or a fully qualified global name (for clients within and outside the cell). The DCE user ID can be up to 256 characters in length.

OS/390-user-ID
Is the OS/390 user ID of the client. All potential DFS server clients must have user IDs on the OS/390 host where the DFS server is running.

DFS-server-name
Is the name that represents a specific DFS server. This is optional. The name of the DFS server is specified in the **_IOE_MVS_SERVER** environment variable for the **dfskern** process. If the mapping entry includes the name of a DFS server, the mapping is valid only for that instance of the DFS server. If the entry does not include the name of a DFS server, the mapping is valid for all servers that are running on the host system that use the Identity Mapping input file (for example, Application Support for Information Management System). The

DFS-server-name for a DFS server is specified in the **_IOE_MVS_SERVER** environment variable in the **dfskern** process.

The first entry type (with three elements) is known as the **specific mapping** to a DFS server. The second entry type (with two elements) is known as the **default mapping** for a DFS server user. One or both of these entry types can be in the Identity Mapping input file.

If a user has both specific mapping and default mapping entries with the same principal in the input file, the specific mapping entry overrides the default mapping entry. The default mapping entries will also no longer apply for other servers.

Each user can only have one default mapping for every DFS principal in the file. Also, each user can only have one specific mapping entry to a particular DFS server for every DFS principal in this file.

Examples

In the following example, the DCE user ID **smith** is mapped to the OS/390 user ID **CMSMITH**. This mapping is effective only when accessing the DFS server whose name is **dfs**. In the second entry, the DCE user ID **jones** is mapped to the OS/390 user ID **TSJONES**. This mapping is effective when accessing **any** DFS server on the host system.

```
smith
CMSMITH
dfs

jones
TSJONES
```

The following example identifies a DCE user ID, **smith**, with both default and specific mapping entries, **CMSMITH** and **TSJOHN**. In this example, the specific mapping entry overrides the default mapping entry.

```
smith
CMSMITH

smith
TSJOHN
dfs
```

Implementation Specifics

This file is part of the OS/390 DFS Base.

The Identity Mapping input file must be created as an HFS file in OS/390 DFS.

Related Information

Command:
mapid

ioepdcf

Purpose

Specifies the processes to be started by the DFS Control Task.

Usage

The **ioepdcf** file, also referred to as the Daemon Configuration File, is an EBCDIC text file used by the DFS Control Task to determine which daemons can be started during the initialization of DFS. The file is located in the **/opt/dfslocal/etc** directory. The information contained in the **ioepdcf** includes the following:

Process Name
The name of the process to be entered in the **ioepdcf** file. Valid processes are:

boserver The **Basic OverSeer Server** (BOS) daemon. The **boserver** daemon monitors all server processes that it starts on the local machine. The **boserver** restarts any failed processes automatically, limiting system outages. For information on processes started by the **boserver** daemon, see “bosserver” on page 538.

butc*nn* The **BackUp Tape Coordinator** (BUTC) daemon. The **butc** server daemon is the Tape Coordinator machine on which backup and restore operations are physically conducted. Valid entries for *nn* are 01 through 08.

export The **export** daemon. The **export** server daemon allows exporting of aggregates.

unexport The **unexport** daemon. The **unexport** server daemon allows unexporting of aggregates.

dfskern The **dfskern** daemon. The **dfskern** server daemon is the DFS File Exporter.

Configuration Type
The configuration type for each process. Available types for OS/390 DFS are:

CONFIGURED=Y
Specifies that the process will start during initialization. If the process abends or ends for any reason, it is automatically re-started by the DFS Control Task process.

Note: Do not use this configuration type for the **export** or **unexport** processes. These processes execute once and stop. The **export** and **unexport** processes must not be automatically re-started once they have run. Use **CONFIGURED=I** or **CONFIGURED=M**.

CONFIGURED=N
Specifies that the process will not be started by the DFS Control Task process nor can the process be started manually.

CONFIGURED=I
Specifies that the process is started during DFS initialization **or** when the **MODIFY** command, **START ALL** is issued. The process may be started manually. The process will not restart if it abends or ends for any reason.

CONFIGURED=M
The process will not be started by the DFS Control Task process but may be manually started by using the OS/390 system command **MODIFY**. The specified process will not restart if it ends for any reason.

Load Module Name

The name of the load module (**LMD**) in a partitioned data set. The load module refers to the name of the member in the *xxx*.SIOELMOD (where *xxx* is installation dependent) data set created during installation (for further information, refer to the *OS/390 Program Directory*).

The following are the PDS member names for the DFS processes started by the DFS Control Task:

boserver The **boserver** daemon load module name. The name, **boserver**, is an alias for the load library entry, **IOEBOSRV**.

butc*nn* The **butc***nn* daemon load module name. There are eight valid load module names for the butc processes: **butc01**, **butc02**, **butc03**, **butc04**, **butc05**, **butc06**, **butc07**, and **butc08**. All are valid aliases for the single load module used for all **butc***nn* processes- **IOEBUTC**

dfskern The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.

In addition, the **export** process is started by the DFS Control Task and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.

Special Parameters

Parameters that are passed to the load module when a daemon is started (including Language Environment/370® (LE/370) runtime options). This is also called the argument list. Any runtime overrides such as storage specifications and redirection of output may also be added. The first argument is the home directory for each process. The home directory points the process to the directory holding the environmental variable (envar) file for the process. Program parameters for the DFS process are preceded with a slash, **/**, in the argument list.

The **ioepdcf** file can be used to override the default parameter options for the following daemons:

boserver The **Basic Overseer Server** daemon monitors all server processes that it starts on the local machine. The **boserver** restarts any failed processes automatically, limiting system outages. See "bosserver" on page 538 for information on options available for this process.

butc*nn* The **Backup Tape Coordinator** daemon is the Tape Coordinator machine on which backup and restore operations are physically conducted. Valid entries for *nn* are 01 through 08. See "butc" on page 540 for information on options available for this process.

export The **export** daemon allows exporting of aggregates. See "dfsexport" on page 597 for information on options available for this process.

unexport The **unexport** daemon allows unexporting of aggregates. See "dfsexport" on page 597 for information on options available for this process.

dfskern The **dfskern** server daemon is the DFS File Exporter. See "fxd" on page 722 for information on options available for this process.

all All of the DFS daemons.

Additional special parameters that control restart and timeout intervals may also be entered in the **ioepdcf** file. These parameters are:

Restart The interval defined for the Control Task to attempt a restart of the process. Restart values are entered in seconds.

Timeout The maximum interval that the Control Task will wait for the process to complete initialization. Timeout values are entered in seconds. If this interval is exceeded with no confirmation of successful completion received by the Control Task, the status of the process is set to **UNKNOWN**.

Examples

The following example is an **ioepdcf** file entry for the dfskern process. The configuration type is **Y**, specifying that the process start during DFS initialization. The load module, **LMD**, is identified as **IOEDFSKN**. In the argument list, **ARG**, **ENVAR** indicates the environment variables to be used. In the example, the home directory is identified as **_EUV_HOME=/opt/dfslocal/home/dfskern**. An LE/370 runtime option is specified: **RPSTG(OFF)**. Parameters for the **fxd** process follow the **/**. The > symbol is a redirection character which indicates that the output is redirected to the DD name that follows. In this example, the redirection of the **STDERR** to **STDOUT** of the process (**dfskern**) is specified by: >**DD:dfskern 2**>**&1**. **RESTART** and **TIMEOUT** values are both set at 300 seconds.

```
dfskern CONFIGURED=Y LMD=IOEDFSKN ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern'),RPSTG(OFF)/
-mainprocs 7 -admingroup subsys/dce/dfs-admin >DD:dfskern 2>&1" RESTART=300 TIMEOUT=300
```

Note: The previous example should be entered on one line even though multiple lines are used in the example.

Implementation Specifics

The **ioepdcf** file is stored as an EBCDIC file in OS/390 DFS.

Related Information

Commands:

bosserver	**dfsexport**	**modify dfs daemon**
butc	**fxd**	**start dfs**

NoAuth

Purpose

Indicates that DFS authorization checking is disabled.

Usage

The **NoAuth** file is a zero-length file that dictates whether DFS authorization checking is enabled or disabled on a server machine. The presence of the **NoAuth** file in the **/opt/dcelocal/var/dfs** directory on a local disk indicates to all DFS server processes on that machine that DFS authorization checking is disabled. All DFS server processes, including the BOS Server, check for the presence of the file when they are requested to perform an operation; they do not check for the necessary administrative privilege for a requested operation when the file is present.

When the **NoAuth** file is present in **/opt/dcelocal/var/dfs** on a server machine, DFS authorization checking is disabled on that machine. The server processes on the machine perform any action for any user who requests it, including the unprivileged identity **nobody**. This is a serious security risk and should be used only in the following situations:

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file.

When the **NoAuth** file is not present in **/opt/dcelocal/var/dfs** on a server machine, DFS authorization checking is enabled on that machine. All DFS server processes on the machine check that the issuer of a command has the proper authorization (is included in the necessary administrative lists) before they perform the requested operation. By default, DFS authorization checking is always enabled on every server machine.

The **bos status** command can be used to determine whether DFS authorization checking is enabled or disabled on a server machine. The command displays the following message if DFS authorization checking is disabled on a machine (it does not display the message if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

The BOS Server on a server machine creates the **NoAuth** file when an authorized user (one listed in the **admin.bos** file on the machine) executes the **bos setauth** command with the **-authchecking** option set to **off** (the file can also be created with the **-noauth** option of the **bosserver** command used to start the BOS Server). The BOS Server removes the file when a user executes the **bos setauth** command with the **-authchecking** option set to **on**. Whenever the **bos setauth** command is used to change the state of DFS authorization checking, all server processes immediately recognize the changed state and respond accordingly to any subsequent commands.

Important Note to Users

On OS/390 DFS, the **bosserver** will erase a pre-existing **NoAuth** file when it starts. Specify the **-noauth** option if you do not want the **NoAuth** file erased.

Cautions

Always use the **bos setauth** command to create the **/opt/dcelocal/var/dfs/NoAuth** file. Do not create the file directly except when explicitly told to do so by instructions for dealing with emergencies (such as server encryption key emergencies). Creating the file directly requires logging into the local operating system of a machine as **root** and using the **touch** command (or its equivalent). On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Implementation Specifics

On OS/390 DFS, the BOS Server will erase a pre-existing **NoAuth** file when it starts. Specify the **-noauth** option if you do not want an existing **NoAuth** file erased.

Related Information

Commands:

bos setauth	**bos status**	**bosserver**

RepLog

Purpose

Contains messages generated by the Replication Server.

Usage

The **RepLog** file contains execution messages and error messages generated by the Replication Server (**repserver** process). The Replication Server runs on every File Server machine in a cell, allowing read-only replicas of filesets to be stored on any File Server machine.

If the **RepLog** file does not already exist when the Replication Server starts, the server process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the Replication Server starts, the process moves the current version of the file to the **RepLog.old** file in the same directory (overwriting the current **RepLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. This file is stored in EBCDIC on OS/390 DFS. It can also be viewed in the same manner as other OS/390 files.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help you evaluate server process failures and other problems.

Implementation Specifics

The **RepLog** file is stored as an EBCDIC file in OS/390 DFS.

Related Information

Commands:

bos getlog	**repserver**

rfstab

Purpose

See "Attributes File (rfstab)" on page 341.

smbidmap

Purpose

Maps an SMB user ID to an OS/390 user ID. The mapping is used to determine the SMB user's corresponding OS/390 user ID. This determines access permissions for shared HFS directories and files and owners for print requests sent to shared printers. If no **smbidmap** file is specified or it does not exist, then the SMB user ID is mapped to the default user ID (specified in the **_IOE_MVS_DFSDFLT** environment variable of DFSKERN). If no default user ID is specified, the request is denied.

Important Note to Users

This command is for SMB support only.

Usage

The **smbidmap** file is a text file that the administrator creates and maintains. This must be created as an HFS file. The location of this file is specified in the **_IOE_SMB_IDMAP** environment variable of DFSKERN.

The smbidmap file contains one or more identity mapping declarations and has the following general format:

SMB-user-ID1 OS/390-user-ID1 ...

This format illustrates an SMB user ID mapping entry. Each entry has two elements: SMB user ID and OS/390 user ID. A blank line is required between entries. The following explains each element in an SMB user ID mapping entry:

SMB-user-ID **or** *Domain/SMB-user-ID* **or** *Workgroup/SMB-user-ID*
: Is the client's SMB identity. This may either be a simple SMB user ID (when you do not care what the domain of the SMB user ID is) or a fully qualified name (for clients within and outside the domain/workgroup). The SMB user ID can be up to 20 characters in length. A Domain/Workgroup name can be up to 15 characters in length.

- *SMB-user-ID* is assumed to be in any domain.
- *Domain/SMB-user-ID* is assumed to be in the specified domain.
- *Workgroup/SMB-user-ID* is assumed to be in the specified workgroup.

OS/390-user-ID
: Is the OS/390 user ID of the client. All potential SMB clients must have OS/390 user IDs on the system where the DFS server is running.

Another entry that is allowed in **smbidmap** is:

```
*
=
```

This means that if no OS/390 user ID can be determined, the SMB user ID should be used as the OS/390 user ID. This only occurs if the SMB user ID is eight characters or less. This entry can be the only entry in the **smbidmap** file, if desired.

Each SMB user can only have one mapping to an OS/390 user ID. However, different SMB users can be mapped to the same OS/390 user ID, if desired.

Examples

In the following example, the SMB user ID **smith** in **domain1** is mapped to the OS/390 user ID **CMSMITH** and SMB user ID **jones** (in any domain) is mapped to the OS/390 user ID **TSJONES**.

```
domain1/smith
CMSMITH

jones
TSJONES
```

Implementation Specifics

The **smbidmap** file is stored as an EBCDIC file in HFS.

smbtab

Purpose

Specifies HFS shared directories and shared printers to be made available to PC clients.

Important Note to Users

This command is for SMB support only.

Usage

The **smbtab** file includes information about each shared directory and each shared printer that can be made available to PC clients. It resides in the directory **/opt/dfslocal/var/dfs**.

The file contains a one-line entry for each shared directory or shared printer. Each entry in the file must appear on its own line.

The shared directory fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab.

Device name
: For shared directories, the device name of the HFS File System that contains the root of the directory path name to be shared; for example, **/dev/ufs2**. This must match the device name of the HFS File System in the **dfstab**.

Share name
: The name to be associated with the HFS directory being shared. A share name can contain numbers (0-9), letters (A-Z) and the following special characters $ % ' - _ @   ! ( ) ¬ # &. To ensure that a client can connect to an OS/390 SMB share, you should limit yourself to these characters. A share name can be up to 12 characters.

  Note: A share name that is longer than 8 characters cannot be connected to by DOS nor by Windows for Workgroup clients.

Device type
: The device type identifier for the type of device housing the share. For HFS directories this must be **ufs**. Enter the identifier in all lowercase letters.

Share description
: The text description of the share. It can be up to 40 characters and is surrounded by double quotes; for example, "Department HFS files".

Shared directories permissions
: For shared directories, specifies whether the share is limited to read-only access or whether read-write access is allowed. Read-only access is specified as **r/o**. Read-write access is specified as **r/w**.

Maximum users
: For shared directories, the maximum number of users that can be connected to a share name. It can be a number between 0 and 4294967295. 0 means that there is no limit to the number of users.

Directory path name
: For shared directories, specifies the path name of the HFS directory (relative to the root of the HFS file system referred to by the Device name) that this share represents. The directory must

reside in a locally mounted HFS file system. This HFS file system must be exported (see "dfstab" on page 372). The directory path name can be up to 1024 characters. It must be surrounded by double quotes if it contains embedded blanks. You may want to export HFS file systems below this directory in order to allow all path names below this directory to be accessible via this share.

The shared printer fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab.

Device name
: For shared printers, the device name of the printer to be shared; for example, `/dev/prt1`.

Share name
: The name to be associated with the printer queue being shared. A share name can contain numbers (0-9), letters (A-Z) and the following special characters $ % ' - _ @ ! ( ) ¬ # &. To ensure that a client can connect to an OS/390 SMB share, you should limit yourself to these characters. A share name can be up to 12 characters.

: **Note:** A share name that is longer than 8 characters cannot be connected to by DOS nor by Windows for Workgroup clients.

Device type
: The device type identifier for the type of device housing the share. For printers this must be prt. Enter the identifier in all lowercase letters.

Share description
: The text description of the share. It can be up to 40 characters and is surrounded by double quotes; for example, "Department printer".

Printer definition name
: For shared printers, specifies the name of the printer definition. The printer definition name is created during the definition of the printer. Refer to the *OS/390 Infoprint Server Operation and Administration* book, S544-5693.

Printer type
: For shared printers, specifies the type of printer. This can be the name of a printer type supplied by Windows. Refer to the *OS/390 Infoprint Server Operation and Administration* book, S544-5693.

When the **dfsshare** command is executed, it reads the **smbtab** file to verify that each HFS directory or printer to be shared is listed in the file. An HFS directory or printer must have an entry in the **smbtab** file if it is to be shared. If a directory or printer is currently shared, it will not shared again. The **smbtab** file is also read and shared directories and shared printers are created during server initialization.

Examples

The following **smbtab** file specifies two HFS directories and one printer should be shared.

```
/dev/ufs2  myshare    ufs  "My share description"  r/w  100  /ghi
/dev/ufs3  hfsshare1  ufs  "Department HFS files"  r/o   50  /project/dept46
/dev/prt1  prt1       prt  "Department printer"    printname1  "Generic / Text Only"
```

Note: You can put comments in **smbtab** file. Comments start with # in column 1.

Implementation Specifics

The **smbtab** file is stored as an EBCDIC file in HFS. An example **smbtab** file can be found in **/opt/dfsglobal/examples**.

Related Information

Commands:

dfsexport	**dfsshare**

Files:

devtab	**dfstab**

TapeConfig

Purpose

Defines configuration parameters for tape drives on a Tape Coordinator machine.

Usage

The **TapeConfig** file includes configuration information about all of the tape coordinators running on a Tape Coordinator machine. A **TapeConfig** file must reside in the directory named **/opt/dcelocal/var/dfs/backup** on each Tape Coordinator machine.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **TapeConfig** file is not necessary on OS/390 DFS and not available.

TE_tapennn

Purpose

Lists error messages from the **butc** process.

Usage

The **TE_***tapennn* file lists error messages generated by the **butc** (Backup Tape Coordinator) process. The **butc** process initializes a tape coordinator on a Tape Coordinator machine (a machine having a tape drive and an associated Tape Coordinator). The **butc** program prompts for new tapes (and displays some additional output) and, if the value set with the **butc** command's **-debuglevel** option is 1, displays information about restore operations on the screen.

The **butc** process also writes error messages to an EBCDIC file named **TE_***tapennn*, where *tapennn* is the Tape Coordinator ID (TCID) (assigned by the **bak addhost** command) with which the process is associated. The file is located in the directory named **/opt/dcelocal/var/dfs/backup** on the local disk of the Tape Coordinator machine. Messages written to the file by the process describe any problems the process encountered while executing an operation; for instance, it can include the names of any filesets the process was unable to include in a dump operation.

Each time the **butc** process is started for a tape drive and tape coordinator pair, it automatically creates the error file. It then appends any messages to the file once it exists. If the file already exists when the **butc** process is started, the process moves the current version of the file to the **TE_***tapennn* **.old** file in the same directory (overwriting the current **TE_***tapennn***.old** file if one exists) before creating a new version to which to append messages. In either case, the issuer of the **butc** command must have write and execute permissions on the **/opt/dcelocal/var/dfs/backup** directory. (The process also writes execution information it generates to the **/opt/dcelocal/var/dfs/backup/TL_***tapennn* file, which it maintains exactly as it does the **TE_***tapennn* file.)

Examples

The following example displays an error file generated by the **butc** process for a tape drive whose device name is dynamically allocated by OS/390 tape services. The file, named **/opt/dcelocal/var/dfs/backup/TE_tape0000** (the log file associated with a tape drive is named **TL_***tapennn*), shows routine error messages generated during a typical execution of the **butc** process. The messages that follow indicate that two dump sets were not added to the Backup Database; messages also indicate why each dump set was not added to the database. The **bak dump** command was used to attempt to add a dump set to the database.

```
Tue Apr 30 10:04:17 1996
: Task 1: dump (dcecell2.yearly2) could not dismount tape Tue Apr 30 10:
1996
: ; status there is not a mounted tape (dfs / btm)
```

Implementation Specifics

The **TE_***tapennn* file is stored as an EBCDIC file on OS/390 DFS.

On OS/390 DFS, there is no need to assign a tape drive to the **butc** process as OS/390 services are used to dynamically allocate tape drives.

Related Information

Command:
butc
File:
TL_*tapennn*

TL_tapennn

Purpose

Lists execution information from the **butc** process.

Usage

The **TL_***tapennn* file is a log file containing execution messages generated by the **butc** (Backup Tape Coordinator) process. The **butc** process initializes a tape coordinator on a Tape Coordinator machine (a machine having a tape drive and an associated tape coordinator). The **butc** program prompts for new tapes (and displays some additional output) and, if the value set with the **butc** command's **-debuglevel** option is 1, displays information about restore operations on the screen.

The **butc** process also writes output to an EBCDIC file named **TL_***tapennn*, where *tapennn* is the Tape Coordinator ID (TCID) (assigned by the **bak addhost** command) with which the process is associated. The file is located in the directory named **/opt/dcelocal/var/dfs/backup** on the local disk of the Tape Coordinator machine. Output written to the file by the process provides information about all operations the process executes, from its startup to its shutdown. The level of detail to which each operation is described depends upon the operation; some operations are described in more detail than others.

Each time the **butc** process is started for a tape drive and tape coordinator pair, it automatically creates the log file. It then appends any messages to the file once it exists. If the file already exists when the **butc** process is started, the process moves the current version of the file to the **TL_***tapennn* **.old** file in the same directory (overwriting the current **TL_***tapennn***.old** file if one exists) before creating a new version to which to append messages. In either case, the issuer of the **butc** command must have write and execute permissions on the **/opt/dcelocal/var/dfs/backup** directory. (The process also writes any error messages it generates to the **/opt/dcelocal/var/dfs/backup/ TE_***tapennn* file, which it maintains exactly as it does the **TE_***tapennn* file.)

Examples

The following example displays a log file generated by the **butc** process for a tape drive dynamically allocated by OS/390 services. The file is named **/opt/dcelocal/var/dfs/backup/TL_tape0000** (the error file associated with this tape drive is named **TL_***tapennn*); it shows routine status messages generated during a typical execution of the **butc** process. The process is executed with the **-debuglevel** set to 0 (zero) on a tape coordinator whose TCID is 0.

```
Tue Apr 30 09:42:51 1996
09:42:51: Starting tape coordinator: TCID 0, debug level: 0, cell /.../
dcecell2.endicott.ibm.com
09:51:46:
09:51:46: Task1: dump (dcedfs1.yearly) Started
09:53:01:
09:53:01: Starting pass 1
09:53:13: Pass 1: Fileset episet2(0,,13) successfully dumped
09:53:23: Pass 1: Fileset episet1(0,,10) successfully dumped
09:53:28: Pass 1: Fileset root.dfs(0,,7) successfully dumped
09:53:28: End of pass 1: Filesets remaining = 0
09:53:28: Task 1: dump (dcedfs1.yearly1) Finished
09:53:28: Task 1: dump (dcedfs1.yearly1) successful : 3 filesets dumped
10:04:17: Task 2: dump (dcecell2.yearly2) No filesets - dump ignored
10:04:17: Task 2: dump (dcecell2.yearly2) could not dismount tape
10:04:17: Task 2: dump (dcecell2.yearly2) Finished
10:04:17: Task 2: dump (dcecell2.yearly2) successful : 0 filesets dumped
10:05:12: Task 3: dump (dcedfs1.monthly1) Started
10:06:07: Task 3: dump (dcedfs1.monthly1) aborted by request
10:06:07: Task 3: dump (dcedfs1.monthly1) : 0 fileset dumps succeeded, 3
fileset dumps failed
```

Implementation Specifics

The **TL_***tapennn* file is stored as an EBCDIC file on OS/390 DFS.

On OS/390 DFS, there is no need to assign a tape drive to the **butc** process as OS/390 services are used to dynamically allocate tape drives.

Related Information

Command:
butc
File:
TE_*tapennn*

UpLog

Purpose

Contains messages generated by the server portion of the Update Server.

Usage

The **UpLog** file contains execution and error messages generated by the server portion (**upserver** process) of the Update Server. The **upserver** process distributes files from the local disk of a machine in response to requests from the client portion (**upclient** process) of the Update Server running on other machines. The **upserver** process should run on the cell's System Control machine and on the Binary Distribution machine for each CPU/operating system type.

Important Note to Users

OS/390 DFS supports only the System Control machine functions. Binary Distribution machine functions are not supported.

If the **UpLog** file does not already exist when the **upserver** process starts, the server process creates the file in the directory named **/opt/dcelocal/var/dfs/adm**. The process then appends any subsequent messages to the file once it exists. If the file exists when the **upserver** process starts, the process moves the current version of the file to the **UpLog.old** file in the same directory (overwriting the current **UpLog.old** file if it exists) before creating a new version to which to append messages.

The process can write different types of output to the file, depending on the actions it performs and any problems it encounters. The file can be viewed with the **bos getlog** command. This file is stored in EBCDIC on OS/390 DFS. It can also be viewed in the same manner as other OS/390 files.

Events are recorded in the log file only at their completion, so the process does not use the file to reconstruct failed operations. However, the contents of the log file can help you evaluate server process failures and other problems.

Note that the **UpLog** file contains execution and error messages for the **upserver** process only; it does not log messages for the **upclient** process. A log file can be specified for use with the **upclient** process when that process is started on a client machine.

Implementation Specifics

The **UpLog** file is stored as an EBCDIC file on OS/390 DFS.

The Binary Distribution machine is not available on OS/390 DFS.

Related Information

Commands:

bos getlog	**upclient**	**upserver**

Vn

Purpose

Contains a chunk of data cached in a disk cache.

Usage

A **V***n* file, or V file, holds a chunk of cached data on a client machine that is using a disk cache. In the name of an actual V file, *n* is an integer; the name of each V file has a unique integer different from other V files on the machine (for example, **V1**, **V2**, and so on). The format of a V file depends on the format of the data it is caching: a V file containing a cached binary file has a binary format; a V file storing a cached ASCII file has an ASCII format.

Each V file always resides in the DCE Local File System aggregate where it's minor device number is specified in the second field of the **CacheInfo** file; it can be overridden to name a different minor device number. The **CacheItems** file records information about each V file, such as its file ID and data version numbers.

The number of V files, or cache chunks, depends on the size of the disk cache (specified in the third field of the **CacheInfo** file, defined with the **dfsd** command's **-blocks** option, or set with the **cm setcachesize** command). For a disk cache, the number of chunks is heuristically computed as the number of cache blocks divided by 8. You can override the default number of chunks with the **dfsd** command using the **-files** option. Specify a positive integer not greater than 32,000.

By default, each V file holds up to 65,536 bytes (64 kilobytes) of a cached file; files larger than 65,536 bytes are divided among multiple V files. A V file can hold only one cached element; if a cached element is smaller than the size of a V file (the chunk size), the remaining space in the V file remains unused.

You can override the default chunk size with the **dfsd** command using the **-chunksize** option. Specify an integer between 13 and 18 to be used as an exponent of 2; the default unit of measure is bytes. For example, a value of 16 equals the default chunk size (2^{16} equals 65,536); a value less than 13 or greater than 18 sets the chunk size to the default, as does a value of 16.

Implementation Specifics

In OS/390 DFS, you cannot access the **V***n* file through OS/390 UNIX. The DCE Local File System aggregate where this file resides is not locally mounted. Always use the commands provided with DFS to alter this files. For further information, see Chapter 13, “Configuring the Cache Manager” on page 251.

Related Information

Commands:	
cm setcachesize	**dfsd**
Files:	
CacheInfo	**CacheItems**

Chapter 19. Distributed File Service Commands

This chapter provides an alphabetical listing of all relevant DFS commands.

Introduction

Purpose

Introduction to the DFS Administration Commands

Usage

Most DFS commands are divided into the following categories, or command suites:

bak	Issued by system administrators to operate the DFS Backup System.
bos	Issued by system administrators to use the Basic OverSeer server (BOS server).
cm	Issued by users to determine machine and cell information; they are used by system administrators to alter and configure the Cache Manager.
fts	Issued by users to check fileset quota information; used by system administrators to manipulate filesets.

In addition, DFS provides a number of miscellaneous commands (for example, **salvage** and **scout**) not associated with a specific command suite.

System administrators use the majority of DFS commands. However, OS/390 DFS users can use the following commands:

- The **cm** commands **cm statservers** and **cm whereis** to determine machine, file, and directory information.
- The **fts** command **fts lsquota** to check quota information.

DFS Command Types

DFS commands follow these general naming rules. Commands beginning with

- **add** (add) or **rm** (remove) affect lists or groups of DFS objects. For example, **bos addadmin** adds an administrative user to an administrative list.
- **cr** (create) and **del** (delete) affect DFS objects. For example, **fts crserverentry** creates a DFS object, a server entry.
- **ls** (list) commands are used to display objects and groups of objects.
- **set** commands are used to assign values to parameters; for example, **fts setrepinfo** assigns replication parameters.
- **get** commands are used to display parameters; for example, **bos getrestart** displays parameters used to automatically restart server processes.

Rules for Using DFS Commands

When supplying an argument to a command, the option associated with the argument can be omitted if:

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax. (The syntax for each command appears with its description in Chapter 19, “Distributed File Service Commands” on page 401.)
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.

- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

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

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

It is also valid to abbreviate a command name to the shortest form that still distinguishes it from the other command names in the suite. For example, it is acceptable to shorten the **bos install** command to **bos i** because no other command names in the **bos** command suite begin with the letter "i". However, there are three **bos** commands that begin with "g": **bos getdates**, **bos getlog**, and **bos getrestart**. To remain unambiguous, they can be abbreviated to **bos getd**, **bos getl**, and **bos getr**.

The following examples illustrate three acceptable ways to enter the same **bos getlog** command:

Complete command:

```
$ bos getlog -server /.../abc.com/hosts/fs1 -file BosLog
```

Abbreviated command name and abbreviated options:

```
$ bos getl -s /.../abc.com/hosts/fs1 -f BosLog
```

Abbreviated command name and omitted options:

```
$ bos getl /.../abc.com/hosts/fs1 BosLog
```

Aliases

An alias is an alternative way of entering an existing command. Each alias is either shorter than the original command, or it is unique within the command's suite (because only the number of characters sufficient to uniquely identify a command need to be entered to execute the command, unique aliases require less typing).

The **bak** suite is the only command suite with aliases. Refer to "bak" on page 408 for a list of the **bak** commands that have aliases.

OS/390 DFS Commands

In OS/390 DFS, most administration and user commands can be run from TSO/E, the OS/390 shell or submitted as a batch job. For a complete list of commands and the corresponding names, if available, for running them in TSO/E, batch and the OS/390 shell, see Table 17 on page 405. For simplicity, the examples in this book are shown in OS/390 shell mode. That is, commands are *entered* from the OS/390 shell The OS/390 shell prompt is shown in examples as the dollar sign ($). In instances where **root** authority is needed (on OS/390 DFS, **root** refers to a user with a **UID = 0**), a number sign (#) is used for the command prompt. There is a slight difference in some command names used to run these facilities when running from TSO/E (or batch) and from OS/390 shell.

While OS/390 DFS command names can be entered in either upper or lowercase in TSO/E (or batch), these commands can only be entered in lowercase in the OS/390 shell.

Note that a percent sign (%) precedes the TSO/E commands for the DFS Utilities: **GROWAGGR**, **NEWAGGR**, **SALVAGE**, and **SCOUT**. This is necessary to allow TSO/E to differentiate the **GROWAGGR** DFS REXX procedure used to support the TSO/E command entry method for these commands from the DFS load module of the same name.

In batch, these names correspond to the partitioned data set (PDS) member names of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) shipped with the OS/390 DFS product for these facilities.

Table 17 lists the user commands and the corresponding names, if available, for running them in TSO/E, batch and the OS/390 shell.

Table 17. IBM OS/390 DFS Commands

Command Description	TSO/E	Batch	OS/390 shell
BAK command	BAK		bak
BOS command	BOS		bos
CM command	CM		cm
DFSEXPORT command	DFSXPORT		dfsexport
FTS command	FTS		fts
GROWAGGR command	%GROWAGGR	GROWAGGR	growaggr
MAPID command	MAPID	IOEMAPID	mapid
NEWAGGR command	%NEWAGGR	NEWAGGR	newaggr
SALVAGE command	%SALVAGE	SALVAGE	salvage
SCOUT command	%SCOUT		scout
UDEBUG command	UDEBUG		udebug

Entering Arguments to OS/390 DFS Commands

Do not enter commands that use all uppercase or mixed-cased arguments from the ISPF command line. When entered from the ISPF command line, arguments that are all lowercase or are of mixed case are converted to all uppercase characters.

If the administrative command has all uppercase or mixed-cased arguments, enter it from the OS/390 shell or from native TSO/E only.

Running the OS/390 DFS Commands in Batch

In batch, DFS command names correspond to the partitioned data set (PDS) member names of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) shipped with the OS/390 DFS product for these facilities.

The names of the PDS members to run batch commands that are shipped with the OS/390 DFS product are listed in the third column of Table 17.

For example, to run the **mapid** command you can do the following:

```
mapid mapid.input mapid.output
```

The following example is of the JCL used to run the **mapid** command:

```
//*jobcard JOB MSGLEVEL=1
//*
//*******************************************************************
//MAPID    EXEC PGM=IOEMAPID,REGION=0M,TIME=1440,
//      PARM=('/DD&colon.INFILE DD&colon.OUTFILE')
//*******************************************************************
//* Parameters
//*******************************************************************
//INFILE   DD PATH='/opt/dfslocal/home/dfskern/mapid.input',
//            PATHMODE=(SIRWXU,SIRGRP,SIXGRP),
//            PATHOPTS=(ORDONLY),PATHDISP=(KEEP)
//OUTFILE  DD PATH='/opt/dfslocal/home/dfskern/mapid.output',
//            PATHMODE=(SIRWXU,SIRGRP,SIXGRP),
//            PATHOPTS=(ORDWR,OCREAT),PATHDISP=(KEEP)
//****************************************************************
//SYSOUT   DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
```

Commands that Cannot Fit in One Line

When entering commands from the OS/390 shell, the command may exceed one line (255 characters). If the command exceeds one line, use the backslash character (\) at the end of the present line to continue to the next line.

Receiving Help

There are several different ways to receive help about DFS commands. The following list summarizes the syntax for the different help options available on OS/390 DFS:

List of commands in a command suite

To view a list of all commands in a command suite, enter the command suite name followed by **help** as follows:

```
$ bos help
```

The command syntax for a single command

To view the syntax of a specific command, enter the suite name, **help**, and the command name, in that order, as follows:

```
$ bos help command_name
```

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

The **apropos** command displays the first line of the online help entry for any command that has a specified string in its name or short description; this is useful if you cannot remember the exact name of a command. If the string is more than a single word, surround it with "" (double quotes) or other delimiters; type all strings in lowercase letters. For example, the following command produces a list of all **bos** commands with the word **create** in their name or short description, as follows:

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

Cautions

Specific cautionary information is included with individual commands.

Privilege Required

The majority of administrative commands require that the issuer's DCE principal or DCE group be included in an **admin** file (for example, **admin.bos**). Some commands require that the issuer have specific permissions to access files (for example, the **delete** permission on a directory) or be logged in as root on the machine where the command is to be issued. On OS/390 DFS, **root** refers to a user with a **UID = 0**. The exact privilege needed to execute a command is detailed with each command.

Implementation Specifics

These commands are part of the OS/390 DFS Base.

In IBM OS/390 DCE Base Services, all commands and file can be run from TSO/E, submitted as a batch job, or run from the OS/390 shell. For simplicity, the commands, files and examples used in this book are shown as entered from the OS/390 shell.

Changes to commands or files for OS/390 DFS are identified in the **Implementation Specifics** sections for the commands or files.

Related Information

Throughout this chapter you will find more information about the commands in a specific suite and a list of the commands in the suite.

Commands:

bak	**cm**	**fts**
bos		

bak

Purpose

Introduction to the **bak** command suite.

Important Note to Users

On OS/390 UNIX services are used to control and dynamically allocate tape drives. The **bak** commands: **bak labeltape**, **bak readlabel**, and **bak scantape** are not necessary on OS/390 DFS and are not available. In addition, the **bak** commands: **bak jobs**, **bak kill**, and **bak quit** can only be entered in interactive mode. For further information, see “Interactive Mode” on page 409.

All **bak** processes apply only to DCE Local File System filesets. No **bak** processes or commands apply to exported OS/390 HFS filesets.

Options

The following options are used with many **bak** commands; they are also listed with the commands that use them:

-server *machine*
: Specifies the file server machine to use with the command. You can use any of the following to specify the File Server machine:

- 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**).

-tapehost *machine*
: Specifies the DCE pathname of the machine (for example, **/.../abc.com/hosts/bak1**) for which a tape coordinator is being added.

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator being used to execute the command. Legal values for this argument are the integers 0 (zero) to 1023. Because the default for the TCID is 0, the drive used most often should be assigned a TCID of 0.

-help
: Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see “Receiving Help” on page 406.

Usage

The **bak** commands are issued by system administrators to work with the DFS Backup System. The commands copy user and system files to backup tapes and restore information from the tapes, if necessary. All **bak** commands are restricted to administrative users only.

The backup system has two main components: a backup database, installed on one or more server machines, and Tape Coordinator machines, which can be any server or client machine. The backup database stores two types of records: dump set records, which list the fileset families and tapes in the dump set; and administrative records, which list fileset families and their entries, dump levels, and tape hosts.

A Tape Coordinator machine must be a machine with a tape drive. It must also run an instance of the **butc**, which is the backup Tape Coordinator process. A *tape coordinator* controls the behavior of its associated drive and accepts service requests from the backup system. A **Tape Coordinator ID** (**TCID**) acts as an identifier for the tape coordinator. The TCID for each Tape Coordinator is assigned in the **TapeConfig** file on the machine that houses the tape drive and in the backup database.

Each TCID is unique to the cell with which the Tape Coordinator is used. With **bak** commands, the TCID specifies the Tape Coordinator to use with the command.

Interactive Mode

The **bak** command suite can be used in regular command mode or in interactive mode. To enter interactive mode, enter **bak** at a command shell prompt. While you are using this mode, the following information applies:

- The word **bak** does not need to be entered with each command; the **bak>** prompt replaces the command shell prompt.
- Regular expression characters do not need to be escaped; in regular command mode, all regular expression characters must be placed in `""` (double quotes) or escaped with a \ (backslash).
- Multiple operations can be tracked with the **bak jobs** command; in regular command mode, pending operations cannot be tracked.
- Currently executing and pending operations can be canceled with the **bak kill** command; in regular command mode, the **bak kill** command cannot be used.
- New connections do not have to be established to the **bakserver** and **flserver** processes, as necessary, each time a command is issued, so execution time is faster than in noninteractive mode.

Descriptions of the **bak jobs**, **bak kill**, and **bak quit** interactive commands follow; interactive commands can be issued *only* in interactive mode (at the **bak>** interactive prompt).

The bak jobs Command: The **bak jobs** command lists the job ID number the backup system has assigned to each dump and restore operation for a tape coordinator; the listed operations can be currently executing or pending. The job ID number is not the same as the unique dump ID number assigned to each dump set by the backup system. (It is also not the same as the task ID number that is sometimes displayed in the output of certain commands; the task ID number can always be safely ignored.)

The complete syntax for the command is

jobs [-help]

The **-help** option prints the online help for the command.

If no operations are executing or pending, the **bak>** prompt returns immediately. Otherwise, the output includes one line for each operation, reporting the following:

- The job ID number.
- A name describing the operation.
- The number of kilobytes transferred so far (from file system to tape for a dump operation, from tape to file system for a restore operation).
- For a dump operation, the string `fileset` followed by the name of the fileset currently being dumped; for a restore operation, the string `fileset` followed by the name of the fileset currently being restored.
- A message indicating the status of the operation. No message is displayed if the operation is executing normally.

The bak kill Command: The **bak kill** command terminates a currently running dump, restore, or tape labeling operation. If the command interrupts a backup operation, all filesets written to the tape before the kill signal is received are complete and usable. The fileset being written when the signal is received may not be complete and *should not be used*. It is best not to use any of the filesets from an interrupted dump.

If the command interrupts a restore operation, all completely restored filesets are online and usable. Because complete restoration of a fileset usually requires data from multiple tapes (a full dump tape and one or more incremental dump tapes), most filesets are usually not completely restored. If the kill signal occurs before the system accesses all of the necessary tapes, most filesets are not restored to the desired date or version and *should not be used*.

If the interrupted restore is overwriting one or more existing filesets, the filesets can be lost entirely; however, the data being restored still exists on tape. In general, to avoid the inconsistencies that can result from an interrupted restore operation, reinitiate the restore operation.

The complete syntax for the command is:

kill -job { *jobID* **|** *dump_set* **} [-help]**

The **-job** option identifies the operation to kill. It can be the following:

- The job ID of the operation, as displayed in the output of the **bak jobs** command.
- The name of the operation, as displayed in the output of the **bak jobs** command if the operation is a dump. Dump set names associated with dump operations have the form *fileset_family_name.dump_level*. It is not possible to distinguish restore operations by name.

The **-help** option prints the online help for the command. All other valid options specified with the **-help** option are ignored.

The bak quit Command: The **bak quit** command exits interactive mode; the regular shell prompt replaces the **bak>** prompt.

The complete syntax for the command is

quit [-help]

The **-help** option prints the online help for the command.

Command and Monitoring Windows

When using the Backup System, you can use a single terminal session as the command window in which to issue **bak** commands to the Tape Coordinators on all Tape Coordinator machines. In addition, you must open a separate monitoring session for each Tape Coordinator process running on a Tape Coordinator machine. The Tape Coordinator process runs in the foreground; any prompts from the Backup System appear in this window.

Aliases

An *alias* is an alternative way of entering a command. Each alias is either shorter than the original command, or it is unique within the command's suite. (Because only the number of characters sufficient to uniquely identify a command need to be entered to execute the command, unique aliases require less typing.)

The **bak** suite is the only command suite with aliases. The following commands in the **bak** suite can also be entered as specified:

bak restoredb
: Can be entered as **bak dbrestore.**

bak restoredisk
: Can be entered as **bak dkrestore**.

bak restoreft
: Can be entered as **bak ftrestore**.

Cautions

Specific cautionary information is included with individual commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ **bak help**
: Displays a list of commands in a command suite.

$ **bak help** *command*
: Displays the syntax for a single command.

$ **bak apropos -topic** *string*
: Displays a short description of any commands that match the specified string.

See “Receiving Help” on page 406 for complete information about the DFS help facilities.

Privilege Required

It is recommended that all system administrators using the backup system be included on the following lists: the **admin.bak** file on all machines that house the backup database; the **admin.fl** file on all machines that house the Fileset Location Database (FLDB); and the **admin.ft** file on all file server machines. The issuer of a **bak** command must be included in the **admin.bak** list on all machines that house the backup database.

Related Information

Commands:
bak adddump
bak addftentry
bak addftfamily
bak addhost
bak apropos
bak deletedump
bak dump
bak dumpinfo
bak ftinfo
bak help
bak job
bak kill
bak labeltape
bak lsdumps
bak lsftfamilies
bak lshosts
bak quit
bak readlabel
bak restoredb
bak restoredisk
bak restoreft
bak restoreftfamily
bak rmdump
bak rmftentry
bak rmftfamily
bak rmhost
bak savedb
bak scantape
bak setexp
bak status
bak verifydb

bak adddump

Purpose

Defines a dump level in the dump hierarchy.

Format

bak adddump -level *dump_level*... **[-expires** *date*...**] [-help]**

Options

-level *dump_level*
: Names each new dump level to be added to the dump hierarchy. Specify a full pathname for each dump level. Precede the name of each level by a / (slash); the / (slash) is a metacharacter that separates each level in a dump level name. When defining a full dump level, precede the name of the level with a / (slash). When defining an incremental dump level, precede the name of each dump level in the name with a / (slash); the elements in the pathname preceding the last one must already exist in the dump hierarchy. The complete pathname of each dump level must be unique within the backup database of the local cell.

Dump level names can have any number of elements. Each element cannot contain more than 28 characters. Complete dump level names cannot contain more than 256 characters. To avoid confusion when dump set names are created, the name should not include a period. When a dump set is transferred to tape, the fileset family name and the last component of the dump level name are joined by a period to form the name of the dump set. Then including regular expression characters, escape each character with a **** (backslash) or "" (double quotes).

-expires *date*
: Defines the expiration date to be associated with each new dump level. Expiration dates can be specified as absolute or relative values. Absolute expiration dates have one of the following formats:

at *mm*/*dd*/*yy* **[***hh***:***mm***] | at** *mm*/*dd*/*yyyy* **[***hh***:***mm***]**

The word **at** is followed by a date (*month*/*day*/*year*) and, optionally, a time (*hours*:*minutes*). When the system creates a dump set at this level, it assigns the specified date as the expiration date of the tape that contains the dump set. Valid values for *yy* are 00 to 37, which are interpreted as the years 2000-2037, and 70 to 99, which are interpreted as 1970-1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038-2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038. For a *yyyy* specification, valid values for *yyyy* are 1970-2069. However, values between 2038-2069 are reduced to 2038.

If specified, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The default time is 00:00 (12:00 a.m.).

If a value is entered for the year (*yy*) that falls between the range of 38 and 69, a random time is generated for the default time.

Relative expiration dates have the format

in [*integer***y] [***integer***m] [***integer***d]**

The word **in** is followed by a number of years (maximum 9999), months (maximum 11), and days (maximum 30), or a combination of these arguments. When the system creates a dump set at this level, it adds the specified values to the current date to calculate the expiration date of the tape that contains the dump set. At least one of the three values must be specified, and the appropriate unit abbreviation (**y**, **m**, or **d**) must always accompany a value. If more than one of the three is specified, they must appear in the order shown. As with absolute dates, a number of years that causes the relative time to extend beyond the year 2038 is effectively truncated to the number of years remaining until 2038.

If you omit this option, tapes created at the specified dump levels have no expiration dates, meaning they can be overwritten by appropriately named dump sets at any time. Although the **-expires** option is followed by an ellipsis, you can specify only one expiration date. (The ellipsis is included to accommodate the DFS command parser).

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak adddump** command defines one or more dump levels in the dump hierarchy that is stored in the backup database and names them as specified by **-level**. Precede each different level in a dump level name by a / (slash) metacharacter. If a dump level is for full dumps, provide only its name preceded by a / (slash) (for example, **/full**).

If a dump level is for incremental dumps, its name resembles a pathname listing the dump levels that serve as its parents, starting with a full dump level and proceeding (in order) down the hierarchy. The dump level's immediate parent (named by the next-to-last element in the pathname) is the reference point that determines which files are included in dump sets made at the dump level. Files with modification time stamps later than the date and time when the volume was dumped at the parent dump level are included.

The optional **-expires** option associates an expiration date with each dump level. The expiration date is applied to tapes containing dump sets made at the dump level; after the specified date, the backup system overwrites the tape's contents with acceptably named dump sets without question.

An attempt to overwrite an unexpired tape fails until the issuer relabels the tape with the **bak labeltape** command. (Because the label records the unexpired expiration date or unacceptable name, erasing the label removes the obstacle to overwriting.) If no expiration date is defined for a tape, the backup system overwrites the dump set on the tape with a dump set of the same name without question. Expiration dates can be either absolute or relative.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command defines a full dump called **/yearly** with a relative expiration date of one year:

```
$ bak addd -level /yearly -expires in 1y
```

The following command defines an incremental dump called **/full/incr1** with a relative expiration date of 3 months and 15 days:

```
$ bak addd -l /full/incr1 -e in 3m 15d
```

The following command defines two dump levels, **week1** and **week2**; both are incremental from the parent, **monthly**, and both are defined to expire at 12:00 a.m. on 1 January 1996:

```
$ bak adddump -l /monthly/week1 /monthly/week2 -e at 01/01/96
```

Related Information

Commands:

bak dump	**bak lsdumps**	**bak rmdump**
bak labeltape		

bak addftentry

Purpose

Defines a fileset family entry in a fileset family.

Format

bak addftentry -family *fileset_family_name* **-server** *machine* **-aggregate** *name* **-fileset** *name* **[-help]**

Options

-family *fileset_family_name*
: Names the fileset family to which this fileset family entry is to be added. The fileset family must already have been created with the **bak addftfamily** command.

-server *machine*
: Indicates the file server machines that house the filesets in the fileset family entry. Legal values for a single machine are the machine's DCE pathname, the machine's host name, or the machine's IP address. You can also specify the regular wildcard expression (**.***), to match all machine names; in noninteractive mode, surround the wildcard with double quotes ("**.***").

-aggregate *name*
: Indicates the aggregates that house the filesets in the fileset family entry. Legal values are the device name or aggregate name of an aggregate (these names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file) or the regular wildcard expression (**.***), which matches any aggregate name. In noninteractive mode, surround the wildcard with double quotes.

-fileset *name*
: Indicates the filesets to be included in the fileset family entry. Legal values are a specific fileset name, the regular wildcard expression (**.***), or a regular expression that includes the regular expression characters described in the Usage section. In noninteractive mode, surround the entire argument with double quotes if it contains regular expression characters or escape each regular expression character with the \ (backslash); otherwise, the command shell attempts to interpret them.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak addftentry** command adds a fileset family entry to the fileset family specified with **-family**. The fileset family must already have been created with the **bak addftfamily** command.

A fileset family entry can include different numbers and groupings of filesets, depending on how the **-server**, **-aggregate**, and **-fileset** options are combined. For the **-server** and **-aggregate** options, the issuer can specify either a single, specific value or the wildcard (**.***). The wildcard matches any string, so it matches every machine name or aggregate name found in the Fileset Location Database (FLDB). The **bak** program initiates a search of the entire FLDB to resolve the wildcards.

For the **-fileset** option, a wider range of notation from the regular expression character set is acceptable and can be combined with specific character strings. Regular expression characters are case-sensitive. In addition to strings of individual letters (which match any occurrence of that exact string) and the wildcard (**.***, which matches any fileset name), the acceptable notation includes the following regular expression characters. Note that these characters cannot be used for server machine or aggregate names.

* (asterisk)
: Matches any number of characters (0 or more) or the previous character and can be combined with any other regular expression character.

[ ] (brackets)
: Around a list of characters match a single instance of any of the characters, but no other characters. For example, **[abc]** matches **a** or **b** or **c** but not **d** or **A** or **ab**. This expression can be combined with the asterisk.

^ (caret)
: When used as the first character in a bracketed set, indicates a match with any single character except the characters that follow it. For example, [^a] matches any single character except lowercase **a**.

. (period)
: Matches any single character, but a character must be present.

\(backslash)
: Can precede any of the regular expression characters in this list so that they match only their literal values. For example, the expression **\\*** matches a single asterisk, and the expression **\\** matches a single backslash.

In the following example, the combination of letters and regular expression characters matches any string that begins with a **user.** prefix and ends with a **.bak** extension:

```
user\..*\.bak
```

The previous example is issued in interactive mode. When issuing this command in noninteractive mode, it is necessary to enclose character strings that include regular expression characters in double quotes or to escape the wildcards with the \ (backslash); for example, **"user\\..*\\.bak"** or **user.\\\\..\\*\\\\.bak** are equivalent to the previous example. Otherwise, the command shell attempts to resolve the regular expression characters rather than pass them to the **bak** command interpreter for resolution. This may result in failure of the command or the creation of incorrect fileset entries.

Possible values for the arguments of the **bak addftentry** command follow. To create a fileset family entry that includes:

- Every fileset in the cell's file system, provide the **.*** wildcard for all three options
- Every fileset on a machine, provide the DCE pathname of the machine with **-server** and the **.*** wildcard for **-aggregate** and **-fileset**
- Every fileset on every aggregate of the same name, provide the aggregate name with **-aggregate** and the **.*** wildcard for **-server** and **-fileset**
- Every fileset in the cell's file system that includes a common string of letters in its name (such as a **.bak** extension), provide the **.*** wildcard for **-server** and **-aggregate** and a character string/regular expression combination for **-fileset**
- Every fileset on one aggregate, provide the DCE pathname of the machine with **-server**, the aggregate name with **-aggregate**, and the **.*** wildcard for **-fileset**
- Every fileset on a specific machine that includes a common string of letters in its name (such as a **.bak** extension), provide the DCE pathname of the machine specified with **-server**, the **.*** wildcard for **-aggregate**, and a character string/regular expression combination for **-fileset**
- Every fileset on each machine's similarly named aggregate that includes a common string of letters in its name (such as a **.bak** extension), provide the **.*** wildcard for **-server**, the aggregate name for **-aggregate**, and a character string/regular expression combination for **-fileset**

- Every fileset on one aggregate that includes a common string of letters in its name (such as a **.bak** extension), provide the DCE pathname of the machine with **-server**, the aggregate name with **-aggregate**, and a character string/regular expression combination for **-fileset**
- A single fileset, provide the DCE pathname of the machine with **-server**, the aggregate name with **-aggregate**, and the fileset name with **-fileset**

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following commands add a fileset family entry that includes all filesets in the cell that begin with a **user.** prefix to the fileset family called **user**. The two commands, issued in noninteractive mode, are equivalent.

```
$ bak addftentry user ".*" ".*" "user\..*"
$ bak addftentry user ".*" ".*" user\\..\*
```

Both of the previous commands could be issued in interactive mode as:

```
bak> addftentry user .* .* user\..*
```

Related Information

Commands:

bak addftfamily	**bak lsftfamilies**	**bak rmftentry**

File:
dfstab

bak addftfamily

Purpose

Creates a new (empty) fileset family in the backup database.

Format

bak addftfamily -family *fileset_family_name* **[-help]**

Options

-family *fileset_family_name*
: Names the new fileset family. The fileset family name must be unique within the backup database of the local cell. It can be no longer than 31 characters, and it can include any characters. (To avoid confusion when dump set names are created, the name should not include a period. When a dump set is transferred to tape, the fileset family name and the last component of the dump level name are joined by a period to form the name of the dump set.)

  Regular expression characters used in the name of the fileset family must be escaped with a \ (backslash) to prevent the command shell from expanding them when working in noninteractive mode; for example, `usr\*` for a fileset family named **usr***. Because they have no meaning in the name of a fileset family, regular expression characters are not recommended.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak addftfamily** command creates a new fileset family in the backup database, assigning it the name specified with the **-family** option. To make it easier to track its contents, the fileset family name should give some indication of the filesets it contains (for example, **user** for the fileset family that includes all user filesets in the file system).

Do not include periods in the fileset family name. The names of tapes that contain dump sets of this fileset family consist of the fileset family name and the final component of the dump level name joined by period.

After issuing this command, enter the **bak addftentry** command to define the fileset entries included in the fileset family. Use the **bak lsftfamilies** command to list the fileset families currently defined in the backup database. Use the **bak rmftfamily** command to remove a currently defined fileset family from the backup database.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Example

The following command creates a fileset family called **sys**:

```
$ bak addftf sys
```

Related Information

Commands:

bak addftentry **bak restoreftfamily** **bak rmftfamily**
bak lsftfamilies

bak addhost

Purpose

Adds a Tape Coordinator entry to the backup database.

Format

bak addhost -tapehost *machine* **[-tcid** *tc_number***] [-help]**

Options

-tapehost *machine*
: Names the machine for which the Tape Coordinator is to be added. Specify the DCE pathname of the machine (for example, **/.../abc.com/hosts/bak1**), its host name (for example, **bak1.abc.com**), or its IP address (for example, **11.22.33.44**).

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) to be assigned to the tape coordinator. Legal values are integers from 0 to 1023. A value must match the TCID assigned to the Tape Coordinator in the **/opt/dcelocal/var/dfs/backup/TapeConfig** file on the **-tapehost** machine, and it must be unique among TCIDs in the backup database of the local cell. Each Tape Coordinator must have its own TCID, but the TCIDs need not be assigned in sequence (for example, it is legal to skip numbers or to assign them out of order). If this option is omitted, a value of 0 (zero) is used.

Important Note to Users

The **TapeConfig** file described in this section is not required by the Backup Tape Coordinator on OS/390 DFS.

Issuing **bak** commands is most convenient if the Tape Coordinator used most often on a Tape Coordinator machine has a TCID of 0 (zero). The **-tcid** option can then be omitted to direct commands to that tape coordinator.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak addhost** command creates an entry for a Tape Coordinator in the backup database. The entry indicates:

- The machine for which the Tape Coordinator is defined (specified by **-tapehost**).
- The tape coordinator's TCID (specified by **-tcid**).

Repeat the command once for each Tape Coordinator on a tape coordinator machine. The backup database allows a maximum of 1024 tape coordinators in the local cell.

The mapping between the TCID of a Tape Coordinator and the device name of the drive with which it is associated is recorded in the **TapeConfig** file on the Tape Coordinator machine. The **TapeConfig** file must be altered accordingly when this command is issued.

Enter the **bak lshosts** command to list the tape coordinators that have entries in the backup database. Enter the **bak rmhost** command to remove the entry for a Tape Coordinator from the backup database.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command creates an entry in the backup database for a Tape Coordinator on the machine named **bak1**. The Tape Coordinator is assigned a TCID of 0 (zero); the mapping between the TCID of the Tape Coordinator and the device name of a tape drive must appear in the **TapeConfig** file.

```
$ bak addhost /.../abc.com/hosts/bak1
```

The following command creates an entry in the backup database for a Tape Coordinator on the machine named **bak2**; the Tape Coordinator has a TCID of 1.

```
$ bak addh /.../abc.com/hosts/bak2 1
```

Related Information

Commands:

bak lshosts	**bak rmhost**

File:
TapeConfig

bak apropos

Purpose

Shows each help entry containing a specified string.

Format

bak apropos -topic *string* **[-help]**

Options

-topic *string*
: Specifies the keyword string for which to search. If it is more than a single word, surround it with "" (double quotes) or other delimiters. Type all strings for **bak** commands in lowercase letters.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak apropos** command displays the first line of the help entry for any **bak** command containing the string specified by **-topic** in its name or short description.

Output

The **bak apropos** command displays the first line of the help entry for any **bak** command containing the string specified by **-topic** in its name or short description. The first line of the online help entry for a command lists the command and briefly describes its function.

To see the syntax for a command, use the **bak help** command.

Examples

The following command lists all **bak** commands containing the word **tape** in their names or short descriptions:

```
$ bak ap tape

labeltape: label tape
readlabel: read label on tape
scantape: list filesets on tape
status: get tape coordinator status
```

Related Information

Command:
bak help

bak deletedump

Purpose

Deletes the record of a dump set from the backup database.

Format

bak deletedump -id *dumpID* **[-help]**

Options

-id *dumpID*
: The dump ID number of the dump set to be deleted form the backup database.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak deletedump** command removes the record of the dump set associated with the specified dump ID from the backup database. It can be used to remove from the backup database the record of a dump that contains incorrect data or for which the corresponding tape is to be discarded.

After the record of a dump set is deleted from the backup database, dump sets for which it serves as the parent, either directly or indirectly, can no longer be used to restore data to the file system. The **bak deletedump** command can be reissued to remove the record of such dumps from the backup database, but leaving a record of them in the database causes no problems. Also, as long as the tape that contains the parent dump set remains available, the **bak scantape** command can be used to restore information about that dump set from the tape to the backup database, again making the dump sets that rely on the parent dump set usable.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak scantape** command is not necessary on OS/390 DFS and not available. The command may be issued against another non-OS/390 Distributed File Service system.

Use the **bak dumpinfo** command to list the dump IDs currently recorded in the Backup Database.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command deletes the record of the dump set with dump ID **653777462** from the backup database:

```
$ bak del 653777462
```

Related Information

Commands:

bak dump	**bak dumpinfo**	**bak scantape**

bak dump

Purpose

Dumps a specific fileset family at a specific dump level.

Format

bak dump -family *fileset_family_name* **-level** *dump_level* **[-tcid** *tc_number***] [-noaction] [-help]**

Options

-family *fileset_family_name*
: Names the fileset family (already defined in the backup database using the **bak addftfamily** and **bak addftentry** commands) to be dumped.

-level *dump_level*
: Indicates the dump level (already defined in the backup database using the **bak adddump** command) to be used in dumping the fileset family. Provide a full pathname for the dump level, including all necessary / (slashes). This option determines whether the dump is full or incremental and, in the latter case, determines which dump level serves as the parent for the dump.

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the tape coordinator for the tape drive containing the tape. If omitted, it defaults to 0 (zero).

-noaction Displays all filesets that would be included in the indicated dump without actually performing the dump. This lets you check a fileset family's size before actually dumping it so that you can calculate the correct number of tapes needed. Specify all other options as you would to actually perform the operation.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak dump** command dumps the fileset family specified by **-family** at the dump level specified as a pathname by **-level**.

- A *full dump* records the structure of all directories in each fileset in the fileset family and includes all the data in each fileset.
- An *incremental dump* also records the structure of all directories in each fileset in the fileset family, but it includes data from only those files in the filesets that changed since the fileset family was dumped at the parent dump level; such files have modification time stamps later than the date and time at which the fileset family was dumped at the parent dump level. The program uses the next-to-last element in the **-level** pathname as the parent dump and consults the backup database to learn the date and time at which this fileset family was last dumped at that level.

  If the program cannot locate a dump set dumped at a parent dump level, it looks recursively in the backup database for a dump set created at the dump level one higher in the pathname. If it can find no dump set created at a higher dump level in the hierarchy, it creates a full dump set.

If the Backup System is unable to access a fileset (for example, because of a file server machine or fileset server outage), it attempts to access the fileset three times over the course of the operation. If it cannot access the fileset after the third attempt, it omits the fileset from the dump instead of stopping the dump entirely. If the Tape Coordinator performing the dump was initialized at debug level 1, a report on the failure to include the fileset appears in the Tape Coordinator's monitoring window. The Tape Coordinator's error file also records the fileset's omission.

If the failure to access a fileset occurs during a full dump, the next incremental dump of the fileset includes the entire fileset. If the failure occurs during an incremental dump, the next incremental dump of the fileset includes all files modified since the fileset was last included in a dump set.

Before writing the dump to tape, the Tape Coordinator checks that the tape in the indicated tape drive has an acceptable name on its label. If the name on the label is not acceptable, the Backup System prompts for the correct tape. There are three acceptable types of names:

- *fileset_family_name.dump_level.index*, where *fileset_family_name* and *dump_level* match the values provided on the command line (with **-family** and **-level**). The *dump_level* is the last component of the specified dump level; the *index* distinguishes this tape from others that contain this same dump set. If a single tape contains the entire dump set, its index is the numeral 1.
- The tape is labeled as empty. The Backup System labels the tape with the correct name of the form *fileset_family_name.dump_level.index*.
- The tape is not labeled because it has never been used in the Backup System. The Backup System labels the tape with the correct name of the form *fileset_family_name.dump_level.index*.

If it finds that the name on the tape label is acceptable, the Backup System checks the expiration date on the tape before it writes data to it. If the expiration date has not expired, the system does not write data to the tape unless the issuer relabels the tape with the **bak labeltape** command (because the label records the expiration date, erasing the label removes the obstacle to overwriting). If the expiration date has expired or if no expiration date is associated with the tape, the system overwrites the contents of the tape without question (given that the tape has an acceptable name).

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak labeltape** command is not necessary on OS/390 DFS and not available. The command may be issued against another non-OS/390 Distributed File Service system.

The tape label also tells the Tape Coordinator the size of the tape. However, the Tape Coordinator applies the capacity specified in the **TapeConfig** file for the tape drive containing the tape to any tape, regardless of the size specified in the tape's label. Make sure the tapes are at least as large as the tape size listed in the **TapeConfig** file. If a tape is larger, some of its capacity simply may not be used for the dump; if it is smaller, the dump may fail, but only after the Backup System fills the tape and determines that the tape is too small for the drive.

In addition, the Backup System checks the expiration date on the tape before it writes data to it. If the date is not expired, the system does not write data to the tape unless the issuer relabels the tape with the **bak labeltape** command (because the label records the expiration date, erasing the label removes the obstacle to overwriting). If the expiration date is expired or if no expiration date is associated with the tape, the system overwrites the contents of the tape without question.

The Backup System does not require that a fileset fit entirely on a single tape. If the Tape Coordinator reaches the end of a tape while dumping a fileset, it puts the remaining data onto the next tape. The backup database automatically records that the fileset is on multiple tapes.

The **-noaction** option instructs the program to display a list of the filesets included in a dump set without actually performing the dump. This allows the issuer to determine how large the filesets are before actually dumping them; the issuer can then better calculate the required number of tapes. The command ignores a value specified with the **-tcid** option if the **-noaction** option is used with the command.

The **bak restoreft**, **bak restoreftfamily**, and **bak restoredisk** commands can be used to restore data dumped with the **bak dump** command. You can use either command to restore data to any type of file system (DCE Local File System or non-Local File System), regardless of the type of file system from which it was dumped. Thus, you can dump and restore data between DCE Local File System and non-Local File System file systems, as well as between different types of non-Local File System file systems. (See **bak restoreft**, **bak restoreftfamily**, and **bak restoredisk** commands for more information about dumping and restoring data between different types of file systems.)

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines. The issuer must also be listed in the **admin.fl** files on all fileset database machines and in the **admin.ft** files on all file server machines from which filesets are to be dumped.

Output

The following header is displayed in the command window followed by a list of the filesets, identified by name and fileset ID number, to be included in the dump set:

```
Preparing to dump the following filesets: list of filesets
```

The following message indicates that the Backup System has passed the dump request to the indicated Tape Coordinator:

```
Starting dump.
```

It is followed by a message that reports the unique dump ID number associated with this dump operation:

```
Dump ID of dump fileset_family_name.dump_level: dump_ID_number
```

The dump ID also appears in the Tape Coordinator monitoring window if the **butc** command is issued with debug level 1. The dump ID is not the same as the job ID number visible with **(bak) jobs** when **bak dump** is issued in interactive mode.

If the issuer includes the **-noaction** option, the output is

```
Starting dump of fileset family 'fileset_family' (dump level 'dump_level')
Total number of filesets: number
Would have dumped the following filesets: list of filesets
```

Examples

The following command dumps the filesets in the fileset family **user** according to the dump level **/full/week2/monday**. The issuer places the necessary tapes in the drive with a TCID of 5.

```
$ bak dump user /full/week2/monday 5

Preparing to dump the following filesets:
user.jones.bak 387623900
user.pat.bak 486219245
user.smith.bak 597315841
       .
       .
Starting dump.
Dump ID of dump user.monday: 34
```

The following command displays the list of filesets to be dumped when the **sys.rs_aix32** fileset family is dumped at the **/full** dump level:

```
$ bak dump sys.rs_aix32 /full -n

Starting dump of fileset family 'sys.rs_aix32' (dump level '/full')
Total number of filesets: 24
Would have dumped the following filesets:
       rs_aix32 124857238
       rs_aix32.bin 124857241
       rs_aix32.etc 124857246
           .          .
           .          .
```

Implementation Specifics

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak labeltape** command is not necessary on OS/390 DFS and not available. The command may be issued against another non-OS/390 Distributed File Service system.

Related Information

Commands:
bak adddump
bak addftentry
bak addftfamily
bak deletedump
bak dumpinfo
bak ftinfo
bak labeltape
bak lsdumps
bak readlabel
bak restoredisk
bak restoreft
bak restoreftfamily
bak rmdump
bak rmftfamily

bak dumpinfo

Purpose

Lists information about specified backups.

Format

bak dumpinfo [{ -ndumps *number* **| -id** *dumpID* **}] [-verbose] [-help]**

Options

-ndumps *number*
: Specifies the number of dumps about which information is to be displayed; information about the most recent number of dumps specified with this option is displayed. If fewer than the specified number of dumps exist, information about all existing dumps is displayed. Use this option or use **-id**; omit both options to list information about the last 10 dumps.

-id *dumpID*
: Specifies the unique dump ID number of a specific dump operation about which information is to be displayed. Use this option or use **-ndumps**; omit both options to list information about the last 10 dumps.

-verbose Includes a detailed list of information about the dump specified with the **-id** option. This option can be used only with **-id**.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak dumpinfo** command lists information about specified dump sets. If a number is specified with **-ndumps**, information about that number of dump sets is displayed (information about the most recent **-ndumps** is displayed); if a specific dump ID number is specified with **-id**, information about that dump is displayed; if both options are omitted, information about the 10 most recent dump sets is displayed.

The command displays information from the backup database. It can be used to display dump IDs prior to using the **bak deletedump** command to delete the record of one or more dump sets from the backup database. To view more detailed information about a specific dump, specify both the **-id** option and the **-verbose** option.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Output

The following information is displayed for each dump listed:

DumpID The dump set's ID number. This is a unique identifier that the backup system uses internally.

parentID The dump ID of the dump set that served as the parent for this dump set. A value of 0 (zero) means this is a full dump set and so has no parent, in which case **lvl** is also 0.

lvl	The location in the dump hierarchy of the dump level used in creating the dump set. A value of 0 indicates a full dump set. A value of 1 or greater indicates an incremental dump set made at the indicated level in the hierarchy.
created	The date and time at which the Backup System started executing the dump operation that created this dump set.
nt	The number of tapes required to record the dump set.
nfsets	The number of filesets included in the dump set.
dump_name	The name of the dump set.

Additional information is displayed if both the **-id** and **-verbose** options are specified.

Examples

The following example displays information about the last three dumps:

```
$ bak dumpinfo -ndumps 3

DumpID     parentID   lvl     created      nt  nfsets  dump_name
----------------------------------------------------------------------
729293644  729289323    1  02/09/93  5:34   1     43    users.tue
729287531  729286818    1  02/08/93  4:52   1     23    users.mon
729286056          0    0  02/07/93  4:27   1     31    users.wk1
```

The following example displays information about a specific dump having an ID number of 823987639:

```
$ bak dumpinfo -id 823987639

Dump: id 823987639,  level 0, filesets 1, created: Sat Feb 10 21:27:18 1996

Tape: name fvt8fam.monthly.1
Filesets 1, created 02/10/96 21:27

 Pos     Clone time    Nbytes Fileset
   1 02/10/96 21:27     12450 epi8set
```

Related Information

Commands:

bak deletedump
bak dump
bak ftinfo
bak lsdumps

bak ftinfo

Purpose

Displays a fileset's dump history from the backup database.

Format

bak ftinfo -fileset *name* **[-help]**

Options

-fileset *name*
: Names the fileset whose dump history is to be displayed. Include a **.backup** extension if the backup version of the fileset (rather than the read-write version) was dumped.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak ftinfo** command displays a dump history for the specified fileset, detailing the dates on which the fileset was cloned (the backup version was made) and dumped and the tapes where it resides. If the dump was made of the backup version, as is usual, then **-fileset** must include the **.backup** extension.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Output

The output lists information about the dump sets in which **-fileset** is included, with the most recent dump set listed first. The output is displayed in six columns, as follows:

DumpID
: The dump set's ID number. This is a unique identifier that the Backup System uses internally. It allows the issuer to check that the parent ID for an incremental dump set matches the dump ID of the dump set created previously.

parentID
: The dump ID of the dump set that served as the parent for this dump set. A value of 0 (zero) means this is a full dump set and so has no parent, in which case **lvl** is also 0 (zero). It normally corresponds to the dump ID of the dump set created previously (the one on the next line of the output).

lvl
: The location in the dump hierarchy of the dump level used in creating the dump set. A value of 0 (zero) indicates a full dump set. A value of 1 or greater indicates an incremental dump set made at the indicated level in the hierarchy.

creation date
: The date and time at which the Backup System started executing the dump operation that created the dump set.

clone date
: The date and time at which the fileset was created. For a backup or read-only fileset, this represents the time at which it was cloned from its read/write source. For a read/write fileset, it indicates when the Backup System accessed the fileset to include it in the present dump set.

tape name
The name of the tape that contains the dump set.

Examples

The following command displays dump information about the fileset named **user.smith.backup**:

```
$ bak ftinfo user.smith.backup

   DumpID parentID lvl creation date  clone date   tape name
654972910 654946323  1 10/01/91 5:07 10/01/90 4:01 users.tuesday.1
654960415 654946323  1 09/30/91 5:11  9/30/90 4:16 users.monday.1
654946323         0  0 09/29/91 5:36  9/28/90 4:31 users.week.1
```

Related Information

Commands:
bak dump **bak dumpinfo** **bak lsdumps**

bak help

Purpose

Shows syntax of specified **bak** commands or lists functional descriptions of all **bak** commands.

Format

bak help [-topic *string...***] [-help]**

Options

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **dump**, not **bak dump**). If this option is omitted, the output provides a short description of all **bak** commands.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak help** command displays the first line (name and short description) of the online help entry for every **bak** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **bak apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

Output

The online help entry for each **bak** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Examples

The following command displays the online help entry for the **bak dump** command:

```
$ bak help dump

bak dump: start dump
Usage: bak dump -family <fileset_family_name> -level <dump_level>
[-tcid <tc_number>] [-noaction] [-help]
```

Related Information

Command:
bak apropos

bak jobs

Purpose

Lists the job ID number assigned by the Backup System to each dump.

Usage

The **bak jobs** command is an interactive-only command that lists the job ID number the Backup System has assigned to each dump and restore operation for a tape coordinator; the listed operations can be currently executing or pending. The job ID number is not the same as the unique dump ID number assigned to each dump set by the Backup System. (It is also not the same as the task ID number that is sometimes displayed in the output of certain commands; the task ID number can always be safely ignored.)

Important Note to Users

The **bak jobs**, **bak kill**, and **bak quit** commands are interactive mode commands. They may be issued *only* in interactive mode at the **bak**> interactive prompt. For further information on using these commands, see “Interactive Mode” on page 409.

Example

When you issue the following:

```
bak dump dfsld /top2 0
```

the status of the **dump** command can be displayed by using the **bak jobs** command:

```
bak> jobs
Job 1: Dump (dfsld.top2)
```

The following example displays the results of the previous dump command when a tape gets mounted.

```
bak> jobs
Job 1: Dump (dfsld.top2), fileset episetd
```

bak kill

Purpose

Ends a currently running dump, restore, or tape labeling operation.

Format

bak> kill {*jobID* **|** *dump_set***}**

Options

jobID	Specifies the unique job ID number of the operation to cancel.
dump_set	Specifies the name of the operation in the form *fileset_family_name.dump_level* if it is a dump.

Note: The *jobID* option is the recommended operation for cancelling. If you use the *dump_set* option there is the possibility of having several instances of the same *dump_set* with unique job numbers assigned to each one. If you issue **kill** *dump_set* the first instance of the specified *dump_set* is canceled, the others are unaffected.

Usage

The **bak kill** command is an interactive-only command that terminates a currently running dump, restore, or tape labeling operation. If the command interrupts a backup operation, all filesets written to the tape before the kill signal is received are complete and usable. The fileset being written when the signal is received may not be complete and *should not be used*. It is best not to use any of the filesets from an interrupted dump.

If the command interrupts a restore operation, all completely restored filesets are online and usable. Because complete restoration of a fileset usually requires data from multiple tapes (a full dump tape and one or more incremental dump tapes), most filesets are usually not completely restored. If the kill signal occurs before the system accesses all of the necessary tapes, most filesets are not restored to the desired date or version and *should not be used*.

If the interrupted restore is overwriting one or more existing filesets, the filesets can be lost entirely; however, the data being restored still exists on tape. In general, to avoid the inconsistencies that can result from an interrupted restore operation, reinitiate the restore operation.

Important Note to Users

The **bak jobs**, **bak kill**, and **bak quit** commands are interactive mode commands. They may be issued *only* in interactive mode at the **bak**> interactive prompt. For further information on using these commands, see “Interactive Mode” on page 409.

Example

The following example displays the **bak kill** command.

```
bak> jobs
Job 1: Dump (dfsld.bottom22)

bak> kill 1
IOEH08580I Job 1: Dump (dfsld.bottom22) Ended due to a Kill request
```

bak labeltape

Purpose

Creates the label on a tape.

Usage

The **bak labeltape** command creates a label, readable by the Backup System, at the beginning of a tape. The issuer can either assign a name with the **-tape** option or omit the option to label the tape as empty.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak labeltape** command is not necessary on OS/390 DFS and not available. This command can be issued against another non-OS/390 Distributed File Service system.

bak lsdumps

Purpose

Displays the dump hierarchy from the backup database.

Format

bak lsdumps [-help]

Option

-help Prints the online help for this command.

Usage

The **bak lsdumps** command displays the dump hierarchy from the backup database. A dump hierarchy can contain more than one full dump level, each of which defines a separate subhierarchy of dump levels. The **bak lsdumps** command displays the multiple subhierarchies if the backup database contains more than one full dump level.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Output

The output depicts the parent/child relationships between full and incremental dump levels in the dump hierarchy. The names of full dump levels are displayed at the far left margin. There can be more than one full dump in the hierarchy; each defines a subhierarchy of dump levels, each of which would presumably be used for dumping different fileset families.

Incremental dump levels are displayed below and indented to the right from their parent dump level, which can be either full or incremental. Incremental dump levels need not be directly below their parent; the amount of indentation alone indicates the parent/child relationship.

Examples

The following example displays a dump hierarchy with a single subhierarchy. It starts with the full dump level **/yearly** and has a expiration date defined as 12:00 a.m. on December 31, 1996.

```
$ bak lsdumps

/yearly  expires at Tue Dec 31 00:00:00 1996
```

Dump sets created at the **/yearly** level are full dumps.

The following example displays a dump hierarchy with two subhierarchies. One subhierarchy starts with the full dump level **/month**, the other with the full dump level **/monday** (their positions at the left margin indicate they are full dump levels).

```
$ bak lsdumps

/month expires in 3m
       /week1
               /tuesday expires in 1m
                       /thursday expires in 1m
       /week2
               /tuesday expires in 1m
                       /thursday expires in 1m
/monday expires in 1m
       /tuesday expires in 1w
               /wednesday expires in 1w
                       /thursday expires in 1w
                               /friday expires in 1w
       /saturday expires at Wed Dec 25 14:20:00 1996
```

In the first subhierarchy, **/month** serves as the parent for the **/month/week1** and **/month/week2** dump levels, as indicated by the indentation (**/month/week2** is an example of how an incremental level need not be directly below its parent). The **/month/week1** dump level serves as the parent for the **/month/week1/tuesday** dump level, which serves as the parent for the **/month/week1/tuesday/thursday** level. These within-week relationships are repeated under **/month/week2**.

Dump sets created at the **/month** level are full dumps. Note that a relative expiration date of three months has been defined for the full dump level **/month**. Dumps performed at the **/month/week1** level include all files modified since the fileset family was dumped at the **/month** level.

Dumps performed at the **/month/week1/tuesday** level include all files modified since the fileset family was dumped at the **/month/week1** level, and dumps done at the **/month/week1/tuesday/thursday** level include all files modified since the dump done at the **/month/week1/tuesday** level.

A relative expiration date of one month is displayed for the **/month/week1/tuesday** and **/month/week1/tuesday/thursday** levels.

Dumps performed at the **/month/week2** level would include all files modified since the fileset family was dumped at the **/month** level. Thus, dumps done at the **/month/week2** level serve as a summary of dumps done since the dump at the **/month/week1** level (they contains all files modified since a full dump was performed at the **/month** level).

Relative expiration dates are shown to be defined at one month for the **/month/week2/tuesday** and **/month/week2/tuesday/thursday** levels.

The second subhierarchy, starting with **/monday**, is similarly constructed. The **/monday** dump level represents a full dump (it is at the far left margin); it is the parent for the **/monday/tuesday** level. A relative expiration date for the **/monday** level is shown to be one month. The **/monday/tuesday** level is the parent for **/monday/tuesday/wednesday**, and so on. A relative expiration expiration date of one week is shown for the **/monday/tuesday**, **/monday/tuesday/wednesday**, **/monday/tuesday/wednesday/thursday**, and **/monday/tuesday/wednesday/thursday/friday** levels.

The **/monday/saturday** level's parent is **/monday**, so dumps performed at that level represent a summary of all the dumps performed at the intervening levels. An absolute expiration date of 2:20 pm on December 25, 1996, is shown for the **/monday/saturday** dump level.

Related Information

Commands:

bak adddump	**bak dumpinfo**	**bak rmdump**
bak dump	**bak ftinfo**	

bak lsftfamilies

Purpose

Lists fileset families defined in the backup database.

Format

bak lsftfamilies [-family *fileset_family_name*] **[-help]**

Options

-family *fileset_family_name*
: Names the fileset family to be displayed with the command. If omitted, all fileset families are displayed.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak lsftfamilies** command displays fileset family entry information about the specified fileset family. If **-family** is omitted, it lists all of the fileset families defined in the backup database. If **-family** is provided, it lists only that fileset family. In both cases, the fileset family entries in each fileset family are displayed.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Output

The output includes a separate entry for each fileset family. The entry defines the fileset family entries that make up the fileset family. Each fileset family entry is assigned an index number; the issuer of the **bak rmftentry** command uses these index numbers to identify the fileset family entries to delete.

Examples

The following command shows the fileset family entries in the two fileset families currently defined in the backup database:

```
$ bak lsftfamilies
Fileset family user:
    Entry 1: server .*, aggregate .*, filesets: user.*\.bak
Fileset family aix31:
    Entry 1: server .*, aggregate .*, filesets: aix31
```

Related Information

Commands:

bak addftentry
bak addftfamily
bak rmftentry
bak rmftfamily

bak lshosts

Purpose

Lists tape coordinator entries in the backup database.

Format

bak lshosts [-help]

Option

-help	Prints the online help for this command.

Usage

The **bak lshosts** command lists the tape coordinator entries currently defined in the backup database. The list includes the tape coordinators defined for all tape coordinator machines in the cell. Each tape coordinator is defined in the backup database and is, by implication, available for use. However, a tape coordinator process does not have to be running on a tape coordinator machine at the time the command is issued for the machine to be displayed with this command.

Enter the **bak addhost** command to add an entry for a tape coordinator to the backup database. Enter the **bak rmhost** command to remove an entry for a tape coordinator from the backup database.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Enter the **bak addhost** command to add an entry for a tape coordinator to the backup database. Enter the **bak rmhost** command to remove an entry for a tape coordinator from the backup database.

Output

The command first displays a `Tape hosts:` header. It then reports the following information for each tape coordinator:

- The name of the machine for which the tape coordinator is defined. (The format of the machine name depends on the form specified by the issuer of the **bak addhost** command.)
- The TCID of the tape coordinator. Valid TCIDs for the tape coordinators on a machine are integers from 0 to 1023.

Examples

The following command displays the tape coordinators currently defined in the backup database:

```
$ bak lshosts

Tape hosts:
Host /.../abc.com/hosts/bak1, TCID 0
Host /.../abc.com/hosts/bak1, TCID 1
Host /.../abc.com/hosts/bak2, TCID 2
Host /.../abc.com/hosts/bak3, TCID 8
Host /.../abc.com/hosts/bak3, TCID 6
Host /.../abc.com/hosts/bak3, TCID 7
```

Related Information

Commands:
bak addhost **bak rmhost**

bak quit

Purpose

Exits interactive mode.

Usage

The **bak quit** command is an interactive-only command that exits interactive mode; the regular shell prompt replaces the **bak**> interactive prompt.

Important Note to Users

The **bak jobs**, **bak kill**, and **bak quit** commands are interactive mode commands. They may be issued *only* in interactive mode at the **bak**> interactive prompt. For further information on using these commands, see “Interactive Mode” on page 409.

bak readlabel

Purpose

Displays the name and size from a tape's label.

Usage

The **bak readlabel** command displays the name and size from the label of the tape in the indicated tape drive. These values are placed on the tape with either the **bak dump** command or the **bak labeltape** command.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak readlabel** command is not necessary on OS/390 DFS and not available. This command can be issued against another non-OS/390 Distributed File Service system.

bak restoredb

Purpose

Restores a backup copy of the backup database.

Format

bak restoredb [-tcid *tc_number***] [-help]**

Alias

bak dbrestore

Options

-tcid *tc_number*
: Specifies the TCID of the tape coordinator for the tape drive from which the backup database is to be restored. If omitted, it defaults to 0 (zero).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak restoredb** command restores a backup copy of the entire backup database. If the backup database becomes damaged, you should delete the old database; then use this command to restore an entirely new version from its backup tape (which must be named **bak_db_dump.1**). The backup database is damaged if the disk housing the database becomes damaged or the **bak verifydb** command fails.

Do not attempt to recover information from a corrupted database. Instead, stop all **bakserver** processes and remove the old backup database from each machine on which it is located.

After the database is removed, restart all **bakserver** processes on the machines on which they were running. Use the **bak addhost** command to add a tape host for the tape coordinator from which you plan to restore the backup database. Next, start the OS/390 DFS **butc** process. Then use the **bak restoredb** command to restore the new version of the database; the **-tcid** option specifies the TCID of the tape coordinator from which to restore the backup database (the tape coordinator just added with the **bak addhost** command).

Use the **bak savedb** command to save the backup database to tape.

Note: When the **bak restoredb** command has successfully completed, the tape host machine used for the **bak restoredb** command may have to be stopped and restarted.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Implementation Specifics

On OS/390 DFS, the **butc** process must be started before the **bak restoredb** command can be used to restore a new version of a backup database.

Related Information

Commands:

bak savedb	**bak verifydb**

bak restoredisk

Purpose

Restores the entire contents of an aggregate.

Format

bak restoredisk -server *machine* **-aggregate** *name* **[-tcid** *tc_number***] [-newserver** *machine***] [-newaggregate** *name***] [-noaction] [-help]**

Alias

bak dkrestore

Options

-server *machine*
Names the file server machine housing the aggregate you want to restore. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name or aggregate name of the aggregate on the indicated **-server** that you want to restore. These names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the tape coordinator for the tape drive in which the issuer is placing the necessary tapes. If omitted, it defaults to 0 (zero).

-newserver *machine*
Names the file server machine to which to restore the data. This is necessary only if it is different from **-server**. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option only if the destination server is different from the server specified with the **-server** option.

-newaggregate *name*
Specifies the device name or aggregate name of the aggregate on **-newserver** to which to restore the data. These names are specified in the first and second fields of the entry for the aggregate in the **dfstab** file. This is necessary only if the aggregate is different from **-aggregate**.

When **bak restoredisk** has completed successfully, while using the **-newserver** and **-newaggregate** options, you need to issue the **fts lsheader -s** *machine* and **fts lsfldb** commands. These commands show the restored filesets that are now located at the **-newaggregate** name on the **-newserver** location. You may delete the old mount points using the **fts delmount** command for these filesets and create new mount points using the **fts crmount** command to access the filesets.

-noaction Specifies that the program is to display the list of tapes necessary to perform the indicated restore but not actually perform the restore.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak restoredisk** command restores the contents of the aggregate specified by **-server** and **-aggregate** to the file system. To do this, the **bak** program contacts the Fileset Location Server (**flserver**) for a listing from the Fileset Location Database (FLDB) of all the filesets that reside on the specified aggregate. It then consults the backup database to learn which tapes contain the full and incremental dumps needed to restore every fileset. This command is useful if a disk or machine crash destroys the data on an entire aggregate.

To restore filesets from the specified aggregate to the same site (the site specified with the **-server** and **-aggregate** options), omit the **-newserver** and **-newaggregate** options. The data in the restored filesets overwrites the filesets' current contents; there is not change in the Fileset Location Database (FLDB) entries for the filesets.

To restore the filesets to an alternate site, include the **-newserver** option, the **-newaggregate** option, or both. The filesets continue to use their existing FLDB entries and fileset ID numbers, and the filesets' FLDB entries are updated to record the new site. The current contents of each fileset are replaced with the data restored from tape. The command allows you to restore filesets to a new site as follows:

- To restore the filesets to a different aggregate on the same File Server machine, specify the new aggregate with the **-newaggregate** option.
- To restore the filesets to an aggregate of the same name on a different File Server machine, specify the new File Server machine in the **-newserver** option.
- To restore the filesets to a completely different site, specify the new File Server Machine with **-newserver** option and the new aggregate with the **-newaggregate** option.

If you specify a new site or aggregate and the filesets to be restored currently exist at their old site, you must use the **fts zap** command to delete the existing filesets before issuing the **bak restoredisk** command. The **bak restoredisk** command fails if you do not use the **fts zap** command to delete the existing filesets before using the **bak restoredisk** command to restore the filesets to the new site.

(Do not use the **fts delete** command to delete the existing filesets and their FLDB entries before issuing the **bak restoredisk** command. If you use the **fts delete** command instead of the **fts zap** command, you cannot use the **bak restoredisk** command to restore the filesets; you can restore the filesets only with the **bak restoreft** command.)

The **-noaction** option instructs the program to produce a list of the tapes the Backup System would need to perform the indicated restore without actually performing the operation. To do so, include the **-noaction** option with all of the other options to be used with the actual command.

Data can be dumped and restored between different types of file systems. For example, data dumped from a DCE Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset; likewise, data dumped from a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System fileset.

Restored data is translated into the appropriate format for the file system to which it is restored. 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 DFS Local File System fileset may be lost if the fileset is restored to a file system that does not support ACLs.

To restore one or more filesets, see “bak restoreft” on page 450. Use the **bak restoreftfamily** command to restore a fileset family or to restore one or more filesets to the same site or to different sites.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all fileset database machines and in the **admin.ft** files on all file server machines to which aggregates are to be restored.

Output

If the issuer does not include the **-noaction** flag, this command returns the unique dump ID number associated with this restore operation. The dump ID is displayed in the command window following the command line and in the tape coordinator's monitoring window if the **butc** command is issued with debug level 1.

The dump ID is not the same as the job ID number visible with the **(bak) jobs** command if this command is issued in interactive mode.

If the issuer includes the **-noaction** option, a list of the tapes necessary to complete the restore operation is displayed. No dump ID number is reported because none is assigned.

Examples

The following command restores the filesets listed in the FLDB as residing on the **lfs2** aggregate of **DCEDFS1** to a new aggregate called **lfs3** on **DCEDFS1**. The **fts zap** command is used to delete existing filesets from the current site before the **bak restoredisk** command is issued. The issuer places the tapes in the drive with TCID 0 (zero).

```
$ bak restoredisk -server /.:/hosts/DCEDFS1 -aggregate lfs2 -newaggregate lfs3 -tcid 0

Total number of filesets : 1

Starting restore
Job 1: Restore finished
```

The following command uses the **-noaction** option to display a list of tapes necessary to restore the filesets listed in the FLDB as residing on the **lfs2** aggregate of **DCEDFS1** to a new aggregate called **lfs3** on **DCEDFS1**. A TCID of 0 (the default) is specified. This command does not actually perform the restore.

```
$ bak restoredisk -server /.:/hosts/DCEDFS1 -aggregate lfs2 -newaggregate lfs3 -tcid 0 -noaction

Total number of filesets : 1

Starting restore

Restore fileset episet1(ID 10) from tape dcedfs1.yearly.1 dump id 83045135
3, position 1.
Restore fileset episet1(ID 10) from tape dcedfs1.apr.1, dump id 830451654, posit
ion 1.
```

Related Information

Commands:

bak dump	**bak restoreftfamily**	**fts zap**
bak restoreft	**fts delete**	

File:

dfstab

bak restoreft

Purpose

Restores filesets from tape.

Format

bak restoreft -server *machine* **-aggregate** *name* **-fileset** *name*... **[-extension** *name_extension*] **[-date** *date*] **[-tcid** *tc_number*] **[-noaction] [-help]**

Alias

bak ftrestore

Options

-server *machine*
: Names the file server machine to which to restore the fileset. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. If the fileset currently exists at a site other than the one specified with this option and the **-aggregate** option, the restored fileset is stored at the specified site and the current fileset is removed.

-aggregate *name*
: Specifies the device name or aggregate name of the aggregate to which to restore the fileset. These names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file. If the fileset currently exists at a site other than the one specified with this option and the **-server** option, the restored fileset is stored at the specified site and the current fileset is removed.

-fileset *name*
: Names each fileset to be restored. Provide the name of the read-write version of each fileset, even if (because of its fileset entry definition in the fileset family) the backup version of a fileset was actually dumped. The command automatically appends a **.backup** extension to the name of a fileset if it can find no record in the Backup Database of a backup performed for the fileset's read/write version.

-extension *name_extension*
: Specifies an extension to add to the restored fileset's name to distinguish it from a fileset of the same name currently existing in the file system. This causes the Backup System to restore the data from tape into a new fileset separate from the existing one. Any string other than **.readonly** and **.backup** is acceptable; if a period is to precede the extension, include it in the string provided.

-date *date*
: Specifies the date prior to which a dump must have been made to be included in the restore. The **-date** option indicates a date-specific restore; only dump sets dated before **-date** are restored.

: If omitted, this option defaults to 0 (zero) and a full restore of the most recently dumped version of the fileset occurs. Otherwise, the legal values have the following formats:

: *"mm/dd/yy [hh:mm]"* | *"mm/dd/yyyy [hh:mm]"*
: This specifies an optional time (*hh:mm*) on the indicated date (*month/day/year*). A value of this type causes a date-specific restore containing only data from dumps

done before the indicated time. The time must be in 24-hour format (for example, **20:30** is 8:30 p.m.). Surround the entire argument with "" (double quotes), if you include the time, because it contains a space; otherwise, the double quotes are not required. If the time is omitted 00:00 (12:00 a.m.) will be specified on the indicated date.

Values that can be interpreted for *yy* run from 00 to 37, which are interpreted as the years 2000-2037, and from 70 to 99, which are interpreted as 1970-1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038-2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038. For a *yyyy* specification, valid values for *yyyy* are 1970-2069. However, values between 2038-2069 are reduced to 2038.

If specified, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The default time is 00:00 (12:00 a.m.).

If a value is entered for the year (*yy*) that falls between the range of 38 and 69, a random time is generated for the default time.

-tcid *tc_number*
Specifies the Tape Coordinator ID (TCID) of the tape coordinator for the tape drive where the issuer is placing the necessary tapes.

-noaction Specifies that the program produce the list of tapes necessary to perform the indicated restore but not actually perform the restore.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak restoreft** command restores the contents of the indicated filesets from tape to the indicated site (file server machine and aggregate).

By default, restores are full, recreating the fileset as it existed when it was last dumped. A full restore includes data from the last full dump and all subsequent incremental dumps (if any). If there are incremental dumps, the issuer is prompted to insert the necessary tapes into the tape drive. To have the program produce a list of the tapes that the Backup System would need to perform the indicated restore without actually performing the operation, include the **-noaction** option.

You can also choose to do a date-specific restore by including the **-date** option. A date-specific restore returns the fileset to the state it was in at its last dump before the indicated date. Rather than including all dumps to the final one done, it includes only the last full dump and any incremental dumps done before the indicated date.

The precise effect of a restore depends on whether the fileset currently exists in the file system and whether you want to preserve its current state.

To overwrite a fileset's current contents with data restored from tape, omit the **-extension** option. The results are as follows:

- If **-server** and **-aggregate** specify the fileset's current site, the restored data overwrites the current contents. There is no change in the Fileset Location Database (FLDB) entry for the fileset.
- If the **-server** and **-aggregate** options specify a new site, the restored data is stored in the new fileset at the indicated site. If you name a new site and the fileset to be restored currently exists at its old site, you must do one of the following before issuing the command:

- Use the **fts zap** command to delete the existing fileset. The fileset continues to use its existing FLDB entry and fileset ID number, and the fileset's FLDB entry is updated to record the new site.
- Use the **fts delete** command to delete the existing fileset and its FLDB entry. The fileset receives a new FLDB entry and a new fileset ID number.

Using the **fts zap** command is the better approach because it preserves the fileset's existing ID number, which allows Cache Managers to continue to access the fileset without updating their tables of mappings between fileset names and fileset ID numbers. The **bak restoreft** command fails if you do not use the **fts zap** or **fts delete** command to delete the existing fileset before using the **bak restoreft** command to restore the fileset to the new site.

To preserve a fileset's current contents but also introduce a restored version into the file system, use the **-extension** option. A new fileset at the site specified with **-server** and **-aggregate** then contains the restored data. It has the same name as the current fileset, with the addition of the distinguishing extension. The fileset server assigns the new fileset a fileset ID number automatically; a new FLDB entry records all the appropriate information about the new fileset.

It is also possible to restore a fileset that no longer has an existing counterpart in the file system. A new fileset at the site specified with **-server** and **-aggregate** contains the restored data.

Data can be dumped and restored between different types of file systems. For example, data dumped from a DCE Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset; likewise, data dumped from a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System fileset.

Important Note to Users

On OS/390 DFS, data cannot be dumped or restored from the non-Local File System.

Restored data is translated into the appropriate format for the file system to which it is restored. 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 DFS Local File System fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Use the **bak restoredisk** command to restore the entire contents of an aggregate. Use the **bak restoreftfamily** command to restore a fileset family or to restore one or more filesets to the same site or to different sites.

Cautions

Overwriting an existing fileset destroys any file created in the current fileset after the date of the last dump included in the restore. It is always safer to preserve the current fileset by using **-extension** to restore data to a new fileset.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all fileset database machines and in the **admin.ft** files on all file server machines to which filesets are to be restored.

Output

If you do not include the **-noaction** option, this command returns the unique dump ID number associated with the restore operation. The dump ID is displayed in the command window directly following the command line and in the tape coordinator's monitoring window if the **butc** command is issued with debug level 1.

The dump ID number is not the same as the job ID number visible with the **(bak) jobs** command if this command is issued in interactive mode.

If you include the **-noaction** option, a list of the tapes necessary to complete the restore operation is displayed. No dump ID number is reported because none is assigned.

Examples

The following command restores the fileset **episet1** to the aggregate named **lfs2** on the File Server machine named **/.:/hosts/DCEDFS1**. The issuer places the tape in the drive with TCID 0 (zero).

```
$ bak restoreft  -server /.:/hosts/DCEDFS1 -aggregate lfs2  -fileset episet1 -tcid 0

Starting restore
Job 1: Restore finished
```

The following command uses the **-noaction** option to display a list of tapes necessary to restore the fileset **episet1** to the aggregate called **lfs2** on the File Server machine named **/.:/hosts/DCEDFS1**. A TCID of 0 (the default) is specified. This command does not actually perform the restore.

```
$ bak restoreft -server /.:/hosts/DCEDFS1 -aggregate lfs2 -fileset episet1 -tcid 0 -noaction

Total number of filesets : 1

Starting restore

Restore fileset episet1(ID 10) from tape dcedfs1.yearly.1 dump id 83045135
3, position 1.
Restore fileset episet1(ID 10) from tape dcedfs1.apr.1, dump id 830451654, posit
ion 1.
```

Implementation Specifics

On OS/390 DFS, data cannot be dumped or restored from the non-Local File System.

Related Information

Commands:

bak dump	**bak restoreftfamily**	**fts zap**
bak restoredisk	**fts delete**	

File:
dfstab

bak restoreftfamily

Purpose

Restores a fileset family or one or more specified filesets from tape.

Format

bak restoreftfamily {-family *fileset_family_name* **| -file** *filename***} [-tcid** *tc_number***] [-noaction] [-help]**

Alias

bak familyrestore

Options

-family *fileset_family_name*
: Names a fileset family to be restored. The command restores all of the filesets in each of the fileset entries in the specified fileset family. See "Using the -family Option" on page 455 for information about using this option. Use this option or use the **-file** option.

-file *filename*
: Specifies the full pathname of a file from which the command is to read the name of each fileset to be restored and the site (File Server machine and aggregate) to which the fileset is to be restored. Specify each fileset

```
machine aggregate fileset
```

See "Using the -file Option" on page 456 for information about using this option. Use this option or the **-family** option.

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the tape coordinator for the tape drive where the issuer is placing the necessary tapes. If omitted, it defaults to 0 (zero).

-noaction Directs the command to produce a list of filesets it would restore without actually restoring the filesets. The command also provides additional information, such as the tapes that contain dumps of the filesets and the sites to which the filesets would be restored. Include the other options as you would to actually execute the command. You can use this option with the **-family** option to write a list of filesets to a file, which you can then modify for use with the **-file** option. See "Output" on page 457 for information about using the **-noaction** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak restoreftfamily** command restores the contents of specified filesets from tape to the file system. The command performs a full restore of each indicated fileset, restoring data from the last full dump and all subsequent incremental dumps (if any) of each fileset. use the **-family** option or the **-file** option to indicate the filesets to be restored, as follows:

- The **-family** option lets you restore all of the filesets included in the fileset entries in a specified fileset family. The command reads the Fileset Location Database (FLDB) to determine the filesets to be restored and restores them to their current sites.

- The **-file** option lets you restore specific individual filesets that have entries in a specified file. The command restores each fileset to the site you specify.

The **-noaction** option instructs the command to produce a list of the filesets it would restore without actually restoring any filesets. The command also provides information about the tapes that contain dumps of the filesets. You can use the **-noaction**option with the **-file** option to determine the tapes required to restore the indicated filesets. You can also use the **-noaction** option with the **-family** option to construct a list of filesets that would be restored with a specific fileset family; you can then modify the list of filesets as necessary to produce a file for use with the **-file** option.

The **bak restoreftfamily** command is useful for recovering from catastrophic losses of data, such as the loss of all filesets on multiple aggregates of a File Server machine or the loss of multiple aggregates from multiple File Server machines. In such cases, the command provides a better approach to recovery than the **bak restoreft** command or the **bak restoredisk** command because:

- It allows you to restore either individual filesets or specialized collections of filesets.
- It allows you to restore different filesets to different sites.

Conversely, the **bak restoreft** command restores one or more filesets to a single site, and the **bak restoredisk** command restores all filesets that reside on a single aggregate to a single aggregate. The **bak restoreftfamily** command provides greater breadth to a restore operation than the other commands that restore data, which provide convenient depth to a restore operation.

Regardless of the command used, data can be dumped and restored between different types of file systems. For example, data dumped from a DCE Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset; likewise, data dumped from a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System fileset.

Important Note to Users

On OS/390 DFS, data cannot be dumped or restored from the non-Local File System.

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 Local File System fileset may be lost if the fileset is restored to a fileset that does not support ACLs.

Using the -family Option

Use the **-family** option of the **bak restoreftfamily** command to restore the filesets included in a fileset family. The command reads the FLDB to determine all filesets that satisfy the fields of the entries in the fileset family. It then looks in the Backup Database to determine the tapes that contain the last full dump and all subsequent incremental dumps of each fileset. It restores each fileset included in an entry in the fileset family to its current site, overwriting an existing version of the fileset.

You can specify the name of an existing fileset family, or you can define a new fileset family and add entries that correspond to the filesets that need to be restored. For example, suppose you need to restore all filesets that reside on the File Server machines named **fs1.abc.com** and **fs2.abc.com.**. You can use the **bak addftfamily** command to create a new fileset family. You can then use the **bak addftentry** command to add the following entries to the new fileset family:

```
/.../abc.com/hosts/fs1  .* .*
/.../abc.com/hosts/fs2  .* .*
```

These entries indicate all filesets on all aggregates on the machines named **fs1.abc.com** and **fs2.abc.com**. Once the new fileset family is defined, you can issue the **bak restoreftfamily** command, specifying the name of the fileset with the command's **-family** option.

In fileset families created for use with the **bak restoreftfamily** command, define entries that match the read/write versions of the filesets. The command automatically appends a **.backup** extension to the name of a fileset if it can find no record in the Backup Database of a backup performed for the read/write version. You can include a **.backup** extension to match the backup versions of filesets only if the backup versions were dumped to tape and the backup versions are still valid in the FLDB entries for the filesets.

Using the -file Option

Use the **-file** option of the **bak restoreftfamily** command to restore each fileset that has an entry in a specified file. The command examines the Backup Database to determine the tapes that contain the last full dump and all subsequent incremental dumps of each specified fileset. It restores each fileset to the site indicated in the specified file. The command does not consult the FLDB.

An entry for a fileset in a file to be used with the command must have the following format:

```
machine aggregate fileset [comments...]
```

The entry provides the following information:

machine
Specifies the File Server machine to which the fileset is to be restored. Identify the machine by its DCE pathname (for example, **/.../abc.com/hosts/fs1**), its host name (for example, **fs1.abc.com**) or its IP address (for example, **11.22.33.44**).

aggregate
Specifies the aggregate to which the fileset is to be restored. Identify the aggregate by its device name (for example, **/dev/lv1**) or by its aggregate name (for example, **lfs1**). These names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

fileset
Specifies the fileset to be restored. Specify the name of the read/write version of the fileset, even if the backup version of the fileset was actually dumped. The command automatically appends a **.backup** extension to the name of the fileset if it can find no record in the Backup Database of a backup performed for the read/write version. (Note that you can specify the name of the backup version of the fileset if the backup version was dumped to tape.)

comments...
All remaining text. The command treats any other text provided with the entry for the fileset as a comment and ignores it. Any additional text is optional.

Do not use wildcards (for example, .*) in any entry. Also, do not include a newline character in an entry for a fileset; each entry must appear on a single line of the file. Include only a single line for each fileset in the file. The command uses only the first line for a given fileset; it ignores all subsequent lines for the fileset.

If you restore a fileset to the site at which it currently exists, the command overwrites the existing version of the fileset. If you restore a fileset to a site other than the site at which it currently exists, you must do one of the following before issuing the command:

- Use the **fts zap** command to delete the existing fileset. The restored fileset continues to use its existing FLDB entry and fileset ID number, and the fileset's FLDB entry is updated to record the new site.
- Use the **fts delete** command to delete the existing fileset and its FLDB entry. The restored fileset receives a new FLDB entry and a new fileset ID number.

Using the **fts zap** command is the better approach because it preserves a fileset's existing ID number, which allows Cache Managers to continue to access the fileset without updating their tables of mappings between fileset names and fileset ID numbers. The **bak restoreftfamily** command fails if you do not use the **fts zap** or **fts delete** command to delete an existing fileset before using the **bak restoreftfamily** command to restore the fileset to a new site.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all fileset database machines and in the **admin.ft** files on all file server machines to which filesets are to be restored.

Output

If you do not include the **-noaction** option, the **bak restoreftfamily** command returns the unique dump ID number associated with the restore operation. The dump ID is displayed in the command window directly following the command line and in the tape coordinator's monitoring window if the **butc** command is issued with debug level 1.

The dump ID number is not the same as the job ID number visible with the **(bak) jobs** command if the **bak restoreftfamily** command is issued in interactive mode. Note that the dump ID and the job ID numbers are not assigned to the operation until the command actually begins to restore filesets.

If you include the **-noaction** option, the command displays on standard output the number of filesets that would be restored, followed by a separate line of information about each fileset. For each fileset, the command provides the following output:

```
machine aggregate fileset fileset_dumped # as fileset_restored; tape tape_name; pos
position_number; date
```

The output provides the following information:

machine The host name of the File Server machine to which the fileset would be restored (for example, **fs1.abc.com**).

aggregate
The aggregate name of the aggregate to which the fileset is to be restored (for example, **lfs1**).

fileset_dumped
The name of the fileset that was dumped (for example, **user.frost**). The command can display the name of the backup version of the fileset (for example, **user.frost.backup**) if that version was dumped.

fileset_restored
The name with which the fileset would be restored (for example, **user.frost**). The command always displays the name of the read/write version of the fileset.

tape_name
The name of the tape that contains the dump set with which the fileset was dumped (for example, **user.full.1**).

position_number
The position of the fileset with respect to other filesets on the tape that contains the dump set (for example, **31**).

date The date and time at which the fileset was dumped (for example, **Wed Jul 13 05:59:01 1994**).

The command displays information only for filesets that have been dumped to tape; for each fileset that has not been dumped, the command displays an error message on standard error output. The command reads the Backup Database to determine everything but the *machine, aggregate*, and *fileset_dumped* information. If you use the **-noaction** option with the **-file** option, the *machine, aggregate*, and *fileset_dumped* information must be provided in the specified file; if you use the **-noaction** option with the **-family** option, the command examines the FLDB to determine this information, so it provides information only for filesets that have entries in the FLDB.

The command displays multiple lines of information for a fileset if one or more incremental dumps were performed since the last full dump of the fileset. The command displays one line of output for the last full dump and one line of output for each incremental dump. It displays the lines in the order in which the dumps would need to be restored, beginning with the full dump. It does not necessarily present all of the lines for a fileset consecutively.

If you intend to write the output of the **-family** and **-noaction** options to a file for use with the **-file** option, include only a single line for each fileset; the command ignores any additional lines for a fileset. You can include any line for the fileset; all lines name the fileset's current site. You do not need to remove the # (number sign) and the information that follows it; the command ignores any characters that follow the third argument on a line.

When the **-noaction** option is included, no dump ID and job ID numbers are reported because none are assigned.

Notes

The amount of time required for the **bak restoreftfamily** command to complete depends on the number of filesets to be restored. However, a restore operation that includes a large number of filesets can take hours to complete. To reduce the amount of time required for the operation, you can execute multiple instances of the command simultaneously, specifying disjoint fileset families with each command if you use the **-family** option, or indicating files that list different filesets with each command if you use the **-file** option. Depending on how the filesets to be restored were dumped to tape, specifying disjoint fileset families can also enable you to make the most efficient use of your backup tapes when many filesets need to be restored.

Example

The following command restores all filesets included in the fileset family named **data.restore**, which was created expressly to restore data to a pair of File Server machines on which all data was corrupted due to a software error. All filesets are restored to the sites recorded in their entries in the FLDB.

```
$ bak restoreftfam data.restore

Starting restore
bak: dump ID of restore operation: 112
bak: Finished doing restore
```

The following command restores all filesets that have entries in the file named **/tmp/restore**:

```
$ bak restoreftfam -file /tmp/restore

Starting restore
bak: dump ID of restore operation: 113
bak: Finished doing restore
```

The file **/tmp/restore** has the following contents:

```
/.../abc.com/hosts/fs1 /dev/lv01 user.abhijit
/.../abc.com/hosts/fs1 /dev/lv01 user.vijay
/.../abc.com/hosts/fs1 /dev/lv01 user.pierette
/.../abc.com/hosts/fs2 /dev/lv01 user.frost
/.../abc.com/hosts/fs2 /dev/lv01 user.wvh
```

Implementation Specifics

On OS/390 DFS, data cannot be dumped or restored from the non-Local File System.

Related Information

Commands:

bak addftentry	**bak restoredisk**	**fts delete**
bak addftfamily	**bak restoreft**	**fts zap**
bak dump		

File:
dfstab

bak rmdump

Purpose

Deletes a dump level from the backup database.

Format

bak rmdump -level *dump_level* **[-help]**

Options

-level *dump_level*
: Names the dump level to be deleted; specify the complete pathname for the dump level to be removed, including any necessary / (slashes).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak rmdump** command deletes the dump level indicated with **-level** from the dump hierarchy in the backup database. If the dump level is a parent, all dump levels that are its children (and their children, if any) are also deleted.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Example

The following command deletes a top level called **week3** from the dump hierarchy:

```
$ bak rmd /week3
```

Related Information

Commands:

bak adddump	**bak dump**	**bak lsdumps**

bak rmftentry

Purpose

Deletes a fileset family entry from a fileset family.

Format

bak rmftentry -family *fileset_family_name* **-entry** *fileset_entry_index* **[-help]**

Options

-family *fileset_family_name*
: Names the fileset family from which to delete the entry.

-entry *fileset_entry_index*
: Identifies the fileset family entry to delete. The legal value is the fileset entry index number, a positive integer. The **bak lsftfamilies** command displays the index number of each fileset family entry in a fileset family (the first entry defined has index 1, the second 2, and so on).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak rmftentry** command deletes the indicated fileset family entry from the fileset family specified with **-family**. Use **-entry** to identify the fileset family entry by its index number.

Cautions

When using the **bak lsftfamilies** command to determine the index number of a fileset family entry to be deleted, note that the index numbers are refreshed after each individual **bak rmftentry** command issued against an individual index number. This renumbers the fileset entries for the fileset family.

For example, a fileset has been determined to have an index number of the fileset family entry of **2** and there are four fileset family entries. Using the **bak rmftentry** command to delete the second fileset family entry will cause the remaining three fileset family index entries to be renumbered 1, 2, and 3.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command deletes the fourth fileset family entry from the fileset family called **sys**. The issuer first used the **bak lsftfamilies** command to determine that the index number of the fileset family entry to be deleted is 4.

```
$ bak rmfte sys 4
```

Related Information

Commands:

bak addftentry

bak lsftfamilies

bak rmftfamily

Purpose

Deletes a fileset family from the backup database.

Format

bak rmftfamily -family *file set_family_name***...** **[-help]**

Options

-family *fileset_family_name*
Names each fileset family to be deleted.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak rmftfamily** command deletes each fileset family specified by **-family** from the backup database. It also deletes the fileset family entries contained in each deleted family. The **bak addftfamily** command is used to add fileset families to the backup database.

Use the **bak lsftfamilies** command to list the fileset families currently defined in the Backup database. Use the **bak rmftentry** command to remove a currently defined fileset family entry from the backup database.

Privilege Required

You must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command deletes the fileset family called **user**:

```
$ bak rmftf user
```

Related Information

Commands:

bak addftfamily	**bak lsftfamilies**	**bak rmftentry**

bak rmhost

Purpose

Removes a Tape Coordinator entry from the backup database.

Format

bak rmhost [-tcid *tc_number*] **[-help]**

Options

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator to be removed. Legal values are integers from 0 to 1023. If this option is omitted, a value of 0 (zero) is used.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak rmhost** command deletes the indicated tape coordinator entry from the backup database. The backup server no longer sends requests to that Tape Coordinator, even if it is still operational on the machine. Repeat this command once for each Tape Coordinator whose entry is to be deleted.

Enter the **bak addhost** command to add an entry for a tape coordinator to the backup database. Enter the **bak lshosts** command to list the Tape Coordinators that have entries in the backup database.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command removes the entry for the Tape Coordinator with TCID 1 from the backup database:

```
$ bak rmhost 1
```

Related Information

Commands:

bak addhost	**bak lshosts**

File:
TapeConfig

bak savedb

Purpose

Creates a backup copy of the backup database.

Format

bak savedb [-tcid *tc_number*] **[-help]**

Options

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for the tape drive to which the database is to be backed up. If omitted, it defaults to 0 (zero).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak savedb** command creates a backup copy of the entire backup database. Designate one tape as the backup database tape; label it with the name **bak_db_dump.1** (it must have this name). The **-tcid** option specifies the TCID of the Tape Coordinator to which to save the backup database; this option is necessary only if the TCID is not 0 (zero).

If the backup database is damaged, delete the old database and use the **bak restoredb** command to restore a new version from tape. Use the **bak verifydb** command to determine if the backup database is damaged.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command backs up the backup database to a tape in the Tape Coordinator with TCID 3:

```
$ bak save 3
```

Related Information

Commands:

bak restoredb	**bak verifydb**

bak scantape

Purpose

Extracts dump set information from a tape.

Usage

The **bak scantape** command reads the tape in the drive controlled by the Tape Coordinator indicated by **-tcid**, extracting information from the tape label and from the fileset header of each fileset on the tape. It does not extract actual data from the filesets, though the information it does extract is needed to restore the data using the backup system.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak scantape** command is not necessary on OS/390 DFS and not available. This command can be issued against another non-OS/390 Distributed File Service system.

bak setexp

Purpose

Sets the expiration date on an existing dump level.

Format

bak setexp -level *dump_level...* **[-expires** *date...***] [-help]**

Options

-level *dump_level*
: Names each dump level whose expiration date is to be set. Provide the full pathname for each dump level, including all necessary / (slashes).

-expires *date*
: Defines the expiration date to be associated with each dump level. Expiration dates can be specified as absolute or relative values. Absolute expiration dates have one of the following formats:

  at *mm/dd/yy*[*hh*:*mm*] **|** **at** *mm/dd/yyyy*[*hh*:*mm*]

  The word **at** is followed by a date (*month*/*day*/*year*) and, optionally, a time (*hours*:*minutes*). Values that can be interpreted for *yy* run from 00 to 37, which are interpreted as the years 2000-2037, and from 70 to 99, which are interpreted as 1970-1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038-2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038. For a *yyyy* specification, valid values for *yyyy* are 1970-2069. However, values between 2038-2069 are reduced to 2038.

  If provided, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.) If omitted, the time defaults to 00:00 (12:00 a.m.).

  If a value is entered for the year (*yy*) that falls between the range of 38 and 69, a random time is generated for the default time.

  Relative expiration dates have the format:

  in [*integer***y] [***integer***m] [***integer***d]**

  The word **in** is followed by a number of years (maximum 9999), months (maximum 11), and days (maximum 30), or a combination of these arguments. At least one of the three must be provided, and the appropriate unit abbreviation (**y**, **m**, or **d**) must always accompany a value. If more than one of the three is provided, they must appear in the order shown. As with absolute dates, a number of years that causes the relative time to extend beyond the year 2038 is effectively truncated to the number of years remaining until 2038.

  If you omit this option, tapes created at the specified dump levels have no expiration dates, meaning they can be overwritten by appropriately named dump sets at any time. Although the **-expires** option is followed by an ellipsis, you can specify only one expiration date. (The ellipsis is included to accommodate the DFS command parser).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak setexp** command sets the expiration date on each dump level specified with **-level**. Each dump level must already exist in the dump hierarchy stored in the backup database.

The expiration date is applied to tapes containing dump sets made at the dump level; after the specified date, the backup system overwrites a tape's contents with acceptably named dump sets without question. The backup system's attempts to overwrite an unexpired tape fail until the issuer relabels the tape with the **bak labeltape** command. (Because the label records the expiration date, erasing the label removes the obstacle to overwriting.) If no expiration date is defined for a tape, the backup system overwrites the dump set on the tape without question.

Expiration dates can be either absolute or relative.

- Absolute expiration dates are defined as a specific month/day/year and, optionally, hours and minutes. A tape with an absolute expiration date expires at that time, regardless of when the dump set on it was created. (If the expiration predates the dump set's creation, the tape is immediately treated as expired.)
- Relative dates are defined as a number of years, months, days, or any combination of the three. When the backup system creates a dump set at the dump level, it calculates the tape's actual expiration date by adding the relative date to the start time of the dump operation.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Examples

The following command associates an absolute expiration date of 10:00 p.m. on 31 December 1995 with the dump level **/95/december**

```
$ bak setexp /95/december -e at 12/31/95 22:00
```

The following command associates a relative expiration date of 7 days with the two dump levels **/monthly/week1** and **/monthly/week2**

```
$ bak set /monthly/week1 /monthly/week2 -exp in 7d
```

Related Information

Commands:

bak adddump	**bak dump**	**bak labeltape**

bak status

Purpose

Reports on the operation that a Tape Coordinator is executing.

Format

bak status [-tcid *tc_number*] **[-help]**

Options

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) of the Tape Coordinator for which status information is to be displayed.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak status** command displays information about the operation currently being performed by the indicated Tape Coordinator. The command displays information about only the Tape Coordinator's current job. It does not display information about any pending jobs waiting for the Tape Coordinator. Use the **jobs** command in interactive mode to display information about the currently executing job and any pending jobs for a Tape Coordinator.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Output

If the indicated Tape Coordinator is not currently performing an operation, the output reports `Tape coordinator is idle`. Otherwise, it reports:

- An operation name describing the operation. One of the following operation names is displayed:

  Dump *dump_set*
  : For a dump operation, where *dump_set* is the name of the dump set in the form *fileset_family_name.dump_level*. Dump operations are initiated with the **bak dump** command.

  Restore
  : For a restore operation. Restore operations are initiated with the **bak restoreft** or **bak restoredisk** command.

  Labeltape *tape_label*
  : For a tape labeling operation, where *tape_label* is the label being placed on the tape. Tape labeling operations are started with the **bak labeltape** command.

  Scantape
  : For a tape scanning operation. Tape scanning operations are initiated with the **bak scantape** command.

  SaveDb
  : For a database saving operation. Operations that save the backup database to tape are started with the **bak savedb** command.

RestoreDb
For a database restoring operation. Operations that restore the backup database from tape are initiated with the **bak restoredb** command.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **bak scantape** command is not necessary on OS/390 DFS and not available. This command can be issued against another non-OS/390 Distributed File Service system.

- The number of kilobytes transferred so far (from file system to tape for a dump operation, from tape to file system for a restore operation).
- The string `fileset` followed by the name of the fileset currently being restored if the operation is a restore; the string `fileset` followed by the name of the fileset currently being dumped if the operation is a dump.
- Status information about the operation. If the operation is executing normally, no message is displayed; otherwise, one of the following messages is displayed:

  [abort request]
  If the **kill** command was issued but the operation is not yet canceled.

  [abort request received]
  If the **status** command was issued and the job is not yet canceled.

  [drive wait] If the Tape Coordinator is waiting for the operator to insert a tape in the drive or waiting for a drive to be free.

  [abort sent] If the operation is canceled but its execution is not yet stopped.

  [operator wait]
  If the Tape Coordinator is waiting for the operator monitoring the operation to insert a tape in the drive.

Example

The following command displays status information about the operation being performed by the Tape Coordinator with TCID 4. The operation is a dump of the dump set whose name is **usersys./monday**. So far, 23,597 bytes have been dumped to tape. The fileset named **user.terry** is currently being dumped. No status message is displayed, indicating the operation is proceeding normally.

```
$ bak status 4

Dump (usersys.monday): 23597 Kbytes transferred, fileset user.terry.
```

Related Information

Commands:

bak dump	**bak restoredb**	**bak restoreftfamily**
bak jobs	**bak restoredisk**	**bak savedb**
bak kill	**bak restoreft**	**bak scantape**
bak labeltape		

bak verifydb

Purpose

Checks the status of the backup database.

Format

bak verifydb [-verbose] [-help]

Options

-verbose Directs the command to provide more information about the backup database.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bak verifydb** command checks the status of the backup database. It displays a message indicating whether the backup database is undamaged or damaged. If the backup database is undamaged, it can be accessed. If it is damaged, it must be restored from tape with the **bak restoredb** command (provided it has been backed up previously with the **bak savedb** command.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bak** files on all backup database machines.

Output

Depending on the condition of the backup database, this command displays one of the following two messages:

`Database can be used.`
Indicates that the database is undamaged and can be used.

`Database is damaged.`
Indicates that the database is damaged. The database must be deleted and then restored from tape.

If the **-verbose** option is included with the command, the command reports some additional information about the backup database. One reason to use the **-verbose** option is to determine if your backup database has any orphan blocks, which are blocks that it preallocated but cannot use. Orphan blocks are not a problem for the database. However, if you are concerned with disk usage on the machine on which the database resides, you can eliminate the unusable blocks by saving the database to tape with the **bak savedb** command and then restoring it with the **bak restoredb** command. (The **-verbose** option also causes the command to display the name of the machine on which the command is issued.

Example

The following command verifies that the backup database is undamaged:

```
$ bak verifydb

Database can be used.
```

Related Information

Commands:

bak dumpinfo	**bak lsdumps**	**bak savedb**
bak ftinfo	**bak restoredb**	

bakserver

Purpose

Initializes the backup server.

Format

bakserver [-adminlist *filename***] [-verbose] [-help]**

Options

-adminlist *filename*
: Specifies the file that contains principals and groups authorized to execute **bakserver** RPCs (usually using **bak** commands). If this option is omitted, the **bakserver** obtains the list of authorized users from the default administrative list file, **/opt/dcelocal/var/dfs/admin.bak**.

-verbose
: Directs the **bakserver** process to provide a detailed report on what it is doing by displaying messages on standard error.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with **bakserver**. See "bos help" on page 504 and "bos apropos" on page 484 for examples of using these commands.

Usage

The backup server (**bakserver** process) communicates with the backup database to perform dump and restore operations. The **bakserver** process must run on all machines that house a copy of the backup database. It is usually started and controlled by the BOS server.

Important Note to Users

On OS/390 DFS, the **bakserver** runs in the DFS address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **bakserver**. For further information about adding or configuring the **bakserver**, refer to "bos create" on page 485.

The first time it is initialized, the **bakserver** creates the backup database in **/opt/dcelocal/var/dfs/backup**; all database files have a root name of **bkdb**. The **bakserver** also creates the **/opt/dcelocal/var/dfs/admin.bak** administrative list file if the file does not already exist.

The principals and groups in the **admin.bak** administrative list are authorized to issue **bak** commands (which are used for tasks such as examining the database and dumping and restoring data). The list must also include the abbreviated DFS server principals of all Backup Database machines to allow for the proper distribution of changes through the Ubik database synchronization mechanism. Because the backup database is a replicated database, the **admin.bak** administrative lists for all **bakserver** processes in a cell must contain the same principals and groups.

It is recommended that all system administrators using the backup system be included on the following lists: the **admin.bak** file on all machines housing the backup database; the **admin.fl** file on all machines housing the Fileset Location Database (FLDB); and the **admin.ft** file on all file server machines.

When it is started, the **bakserver** makes a **Ubik_ServerInit** call to register its existence as a server process with the Ubik coordinator. It then listens for incoming RPCs to which to respond.

When it is started, the **bakserver** also creates the **/opt/dcelocal/var/dfs/adm/BakLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when the **bakserver** is started, the process moves it to the **BakLog.old** file in the same directory (overwriting the current **BakLog.old** file if it exists) before creating a new version to which to append messages.

Privilege Required

The issuer must be logged in as root on the local machine.

Output

If problems are encountered during initialization, the **bakserver** prints error messages on standard error output. The **bakserver** keeps an event log file in **/opt/dcelocal/var/dfs/adm/BakLog**. If run with the **-verbose** option, the **bakserver** process provides a detailed report on what it is doing by displaying messages on standard error.

Implementation Specifics

On OS/390 DFS, the **bakserver** command applies **only** to Local File System filesets. The **bakserver** process does not apply to exported OS/390 HFS filesets or RFS filesets.

Related Information

Files:

admin.bak	**BakLog**

bos

Purpose

Introduction to the **bos** command suite.

Options

The following options are used with many **bos** commands. They are also listed with the commands that use them.

-server Names the machine running the BOS server that is to execute the command. If you want 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 file server machine using the full DCE pathname, for example, **/.../abc.com/hosts/fs1**).

If you want to run an unprivileged **bos** command, 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**).

Note: If you run a **bos** command and specify the file server machine with either the machine's host name or IP address, 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 warning message (unless the **-noauth** flag is used to suppress it):

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

-noauth Directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. Generally, the **-noauth** option is included with a command if DFS authorization checking is disabled on the server machine whose BOS server is to execute the command or if the Security Service is unavailable. If DFS authorization is disabled, the BOS server requires no administrative privilege to issue any command; any user, even the identity **nobody**, has sufficient privilege to perform any operation. If the Security Service is unavailable, a user's security credentials cannot be obtained.

DFS authorization is disabled with the **bos setauth** command or by including the **-noauth** option when the **bosserver** process is started on a machine. DFS authorization is typically disabled

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file.

Include the **-noauth** option with a command that requires administrative privilege only if DFS authorization checking is disabled on the necessary machine. A command that requires administrative privilege fails if the **-noauth** option is included and DFS authorization checking is not disabled. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example **/.../abc.com/hosts/fs1/dfs-server**. (Do not confuse a machine's DFS server principal with its unique **self** identity.)

Use this option only if the command is issued from a DFS server machine. You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored. For complete details about receiving help, see "Receiving Help" on page 406.

Usage

The **bos** commands are used by system administrators to contact the Basic OverSeer (BOS) server. The BOS server runs on every DFS server machine to monitor the other server processes on the machine. It restarts processes automatically if they fail. The BOS server also provides an interface through which system administrators can start and stop processes and check on server status.

The files described in the following sections are used to store configuration, administrative, and security information.

The bos Key Management Routines and Encryption

If a **bos**: key management command, such as **addkey**, **gckeys**, **genkey**, **lskeys**, or **rmkey**, is issued from a system that has a stronger encryption level installed than the encryption level of the server referenced by the command, the key management command will fail. The error typically seen is the following:

```
bos: communications failure (dce / rpc) error encountered while listing
keys
```

To avoid this problem, install the same encryption level on every system in the cell, or issue the key management command from a system that has the same or lesser encryption level as the server referenced.

The bos admin Commands and Principal Name Caching

The DCE Security system maintains a cache of mappings between principal names and UUIDs for each authenticated process. The mapping can cause confusion in the event that you have deleted and recreated a principal with the same name and then used the principal in the **bos addadmin**, **bos lsadmin**, or **bos rmadmin** command. Since the **bosserver** may have this principal in its name cache, it will add the incorrect UUID to the administrative list.

To avoid this problem, restart the **bosserver** on the machine where you are manipulating administrative lists after recreating the principal. This establishes a new cache. This is particularly important if you have deleted and recreated the principal **hosts/dce_hostname/self** or **hosts/dce_hostname/dfs-server**.

The BosConfig File

The **/opt/dcelocal/var/dfs/BosConfig** file on the local disk of each file server machine contains information about the processes the BOS server is to monitor. This information includes the process type, the command parameters associated with the process, and a status flag that tells the BOS server to start the process at initialization or restart the process if the process fails. Whenever the BOS server starts or restarts, it reads the file to learn which processes to monitor; this information is transferred into memory and the file is not read again until the BOS server next restarts.

The administrator can change the process status in the BOS server's memory with specific **bos** commands; therefore, it is possible for a process to stop running even if its status flag in the **BosConfig** file is set to **Run**. Similarly, an administrator can start a process without setting its status flag in the **BosConfig** file to **Run** by changing its memory state flag to **Run**.

Never edit the **BosConfig** file directly; always use the appropriate **bos** commands. Editing the file directly may introduce changes of which the BOS server is unaware. The BOS server does not recognize such changes until it is restarted and again reads the file.

The admin.bos File

The **/opt/dcelocal/var/dfs/admin.bos** file on the local disk of each file server machine contains the names of users who are allowed to issue **bos** commands on that machine. All users can list the contents of the file with the **bos lsadmin** command; only administrative users can edit the contents of the file with the **bos addadmin** and **bos rmadmin** commands. Because the **admin.bos** file is a binary file, you cannot edit it directly; you must use the appropriate **bos** commands.

The Keytab File

The **/krb5/v5srvtab** keytab file is stored on the local disk of each file server machine. A keytab file contains the list of server encryption keys used by a server process on that machine to decrypt tokens presented by clients. The server process interacts only with clients possessing tokens encrypted with server encryption keys listed in the appropriate keytab file.

The keys in a keytab file are marked with a unique key version number. All tokens presented by clients are also marked with a key version number; a server process uses the key version number to determine which key to use to decrypt a token.

Only administrative users can examine, add, and remove keys in the keytab file. Never edit a keytab file directly; always use the appropriate **bos** commands.

Note: The **/krb5/v5srvtab** keytab file is a file included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options available on OS/390 DFS:

$ **bos help**
Displays a list of commands in a command suite.

$ **bos help** *command*
Displays the syntax for a single command.

$ **bos apropos -topic** *string*
Displays a short description of any commands that match the specified *string*.

See "Receiving Help" on page 406 for more information.

Cautions

Never directly edit a **BosConfig** file, a keytab file, an **admin.bos** file, or any administrative (**admin**) file; always use the appropriate commands from the **bos** command suite.

Privilege Required

All **bos** commands can be issued by users listed in the **admin.bos** file on the machine whose BOS server is executing the command. Specific privilege information is listed with each command's description. In addition, if the BOS server is running with DFS authorization checking disabled, no privilege is required to issue **bos** commands.

Implementation Specifics

The following **bos** commands are not supported in OS/390 DFS when issued against the OS/390 DFS **bosserver**. These commands may be issued from OS/390 against other non-OS/390 **bosserver** processes.

- **bos getdates** command
- **bos install** command
- **bos prune** command
- **bos uninstall** command.

Related Information

Commands:

bos addadmin	**bos help**	**bos setauth**
bos addkey	**bos install**	**bos setrestart**
bos apropos	**bos lsadmin**	**bos shutdown**
bos create	**bos lscell**	**bos start**
bos delete	**bos lskeys**	**bos startup**
bos gckeys	**bos prune**	**bos status**
bos genkey	**bos restart**	**bos stop**
bos getdates	**bos rmadmin**	**bos uninstall**
bos getlog	**bos rmkey**	**dcecp**
bos getrestart		

Files:

admin.bak	**admin.ft**	**BosConfig**
admin.bos	**admin.up**	**/krb5/v5srvtab**
admin.fl		

Note: The **/krb5/v5srvtab** keytab file and the **dcecp** command are included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

bos addadmin

Purpose

Adds a user, group, or server to an administrative list.

Format

bos addadmin -server *machine* **-adminlist** *filename* **[-principal** *name*...**] [-group** *name*...**] [-createlist] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
Names the server machine that houses the administrative list to which principals, groups, or both are to be added. The BOS server on this machine executes the command. 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.

-adminlist *filename*
Names the administrative list to which principals, groups, or both are to be added. The complete pathname is unnecessary if the list is stored in the default configuration directory (**/opt/dcelocal/var/dfs**).

-principal *name*
Specifies the principal name of each user or server machine to be added to the administrative list. A user from the local cell can be specified only by a full or abbreviated principal name (for example, **/.../***cellname*/*username* or just *username*) a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, **/.../***cellname*/**hosts**/*hostname*/**self** or just **hosts**/*hostname*/**self**); a server machine from a foreign cell can be specified only by a full principal name.

-group *name*
Specifies the name of each group to be added to the administrative list. A group from the local cell can be specified by a full or abbreviated group name (for example, **/.../***cellname*/*group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

-createlist
Specifies that the file indicated with **-adminlist** is to be created if it is not found. Any principals or groups specified with the command are added to the new file; if no principals or groups are specified, the command creates an empty file. This option has no effect if the specified file already exists.

Note: Because the **admin.bos** list must already exist to issue this command, this option is ignored if **admin.bos** is specified with the **-adminlist** option.

-noauth Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs bos to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos addadmin** command adds the specified users, groups, and servers to the administrative list specified by the **-adminlist** option on the server machine indicated by the **-server** option. The principal (login) names of users and the principal names of server machines to be added to the administrative list are specified with the **-principal** option; the names of groups to be added to the list are specified with the **-group** option. Principals added to the administrative list either directly (with the **-principal** option) or indirectly (as members of groups indicated with the **-group** option) can then issue administrative commands for the DFS server process associated with the list.

The default path for administrative lists is the configuration directory (**/opt/dcelocal/var/dfs**). If the specified list is stored in the default directory, only the specific filename is required. If the specified list is stored elsewhere, the pathname to the file that was used when the associated server process was started is required.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following command adds the user names **jones** and **smith** to the **admin.bos** file on **fs1**. The administrative list is stored in the default configuration directory.

```
$ bos adda -server /.../abc.com/hosts/fs1 -adminlist admin.bos -principal jones smith
```

Related Information

Commands:

bos lsadmin	**bos rmadmin**	

Files:

admin.bak	**admin.fl**	**admin.up**
admin.bos	**admin.ft**	

bos addkey

Purpose

Converts a string into a server encryption key and adds it to a keytab file.

Format

bos addkey -server *machine* **-kvno +***or_version_number* **-password** *string* **[-principal** *name***] [-localonly] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
Names the server machine whose keytab file is to have a new key added to it. The BOS server on this machine executes the command. 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.

-kvno *+or_version_number*
Defines the key version number of the new key. The version number must be one of the following:

- An integer in the range 1 (zero) to 255. The command uses the specified number as the version number of the new key. The integer must be unique for the principal specified by **-principal** in the keytab file on the machine specified by **-server**.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer is unique for the principal indicated by **-principal** in the Registry Database. However, it may not be unique for the indicated principal in the keytab file on the specified machine, in which case it replaces the key currently associated with the principal/version number pair in every keytab file.

Unless the **-localonly** option is used, the new key and its version number replace the key and version number currently stored in the Registry Database for the indicated principal.

-password *string*
Defines a character string to be converted into an octal string for use as the key. The string serves as a password for the indicated principal. It can include any characters; it can also include spaces if the entire string is enclosed in "" (double quotes).

-principal *name*
Provides the principal name with which the key is to be associated. The default is the DFS principal name of the machine specified by **-server**.

-localonly
Specifies that the key is to be added to the keytab file on the machine indicated by **-server**, but that the Registry Database is not to be updated. The default is both to add the key to the local keytab file and to update the Registry Database accordingly.

-noauth Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos addkey** command associates a new server encryption key with the principal name indicated by **-principal** in the **/krb5/v5srvtab** keytab file on the machine specified by **-server** and, by default, in the Registry Database. The key is derived from the string specified by **-password** and is given the version number specified by **-kvno**. The issuer can either specify a version number or have the command choose one that is unique for the indicated principal in the Registry Database. If the **-localonly** option is omitted, the server encryption key and version number for the indicated principal are automatically updated both in the keytab file on the specified server machine and in the Registry Database; if the **-localonly** option is specified, the keytab file is updated, but the Registry Database is not.

Note: The **/krb5/v5srvtab** keytab file is a file included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

The **bos genkey** command is a more secure way of adding a key to a keytab file because it generates a random key. It also always updates the Registry Database. The keytab file must already exist before the **bos addkey** or **bos genkey** commands can be used to add a key to it. (Keytab files are created using the **dcecp keytab create** command.)

If the **bos rmkey** command is issued against the key that is currently being stored in the registry, the passwords get out of sequence (key's password in the local keytab file versus key's password in the registry). This happens because the **bos rmkey** command does not affect the registry as the **bos genkey** and **bos addkey** commands do when they are not used with the **-localonly** option. The following **rgy_edit** command can be used to get back into sequence:

```
rgy_edit> ktdelete -p <affected principal> -v <xx>

(do this for every xx version in the keytab file)

rgy_edit> do a
rgy_edit> change -p <affected principal> -g <group> -o <org -pw <newpw> -mp <ce
ll_admin's pw>
rgy_edit> ktadd -p <affected principal> -pw <newpw>
```

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**, and, unless the **-localonly** option is used, the DFS server principal of the machine specified by **-server** must have the permissions necessary to alter entries in the Registry Database.

Output

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

Example

The following command adds a new server encryption key with key version number **14** to the keytab file on **fs1** without updating the Registry Database. Because **-principal** is omitted, the key is associated with the DFS principal name of **fs1** (the machine specified with **-server**). The password string **"fourteenth new key"** is converted into an octal key before being placed in the keytab file.

```
$ bos addk /.../abc.com/hosts/fs1 14 "fourteenth new key" -localonly
```

Related Information

Commands:

bos gckeys	**bos lskeys**	**dcecp**
bos genkey	**bos rmkey**	

File:
/krb5/v5srvtab

Note: The **/krb5/v5srvtab** keytab file is a file included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

bos apropos

Purpose

Shows each help entry containing a specified string

Format

bos apropos -topic *string* **[-help]**

Options

-topic *string*
Specifies the keyword string for which to search. If it is more than a single word, surround it with " " (double quotes) or other delimiters. Type all strings for **bos** commands in lower case.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos apropos** command displays the first line of the help entry for any **bos** command containing the string specified by **-topic** in its name or short description.

Privilege Required

No privileges are required.

Output

The first line of the online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **bos** command where the string specified by **-topic** is the part of the command name or the first line.

To see the syntax for a command, use the **bos help** command.

Example

The following command lists all **bos** commands that have the word **restart** in their names or short descriptions:

```
$ bos apropos restart

getrestart: get restart times
restart: restart all processes
setrestart: set restart times for server processes
```

Related Information

Command:
bos help

bos create

Purpose

Creates a new process in the **BosConfig** file and starts it.

Format

bos create -server *machine* **-process** *server_process* **-type** *process_type* **-cmd** *cmd_line***...**
[{-noauth | -localauth}] [-help]

Options

-server *machine*
Names the server machine on which to create the new process. The BOS server on this machine executes the command. 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 **name** with either the machine's host name or IP address.

-process *server_process*
Names the server process to be created. You can choose any name for a process, but it is recommended that you give the process the same name as its binary file (and use the same name on every machine running that process). The recommended names are:

ftserver	For the Fileset Server process.
flserver	For the Fileset Location Server process
upclient	For the client portion of the update server, which brings common configuration files and binary files from the System Control machines.
upserver	For the server portion of the Update Server process.
repserver	For the Replication Server process.
bakserver	For the Backup Server process.

Each process runs under the local identity **root** and the DCE identity **self**. However, the process is unauthenticated as far as DFS is concerned. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Important Note to Users

If you plan to run the **bakserver**, **ftserver**, **repserver**, **upclient**, or **upserver** processes under OS/390, be sure to read the appropriate information for the processes. See “bakserver” on page 473, “ftserver” on page 720, “upclient” on page 765, “upserver” on page 768, or “repserver” on page 745 for detailed information.

-type *process_type*
Specifies the process type. Legal values are **simple** and **cron**. Specify **simple** for continuous processes and **cron** for processes that are to run only at specified times.

-cmd
Specifies the commands the BOS server runs to start the process and, if **-type** is **cron**, the time the BOS server executes the command.

For a **simple** process, this must be the complete pathname to the binary file for the process (for example, **/opt/dfsglobal/bin/flserver** for the Fileset Location Server). The commands for some **simple** processes take options, in which case the entire argument must be surrounded by double quotes.

Important Note to Users

On OS/390 DFS, **simple** processes are not identified by complete pathnames to the binary files for the processes. All **simple** process commands on OS/390 DFS are members of the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) created during installation (for further information, refer to the *OS/390 Program Directory*). As all OS/390 member names are limited to eight or less characters, the **bakserver** and **repserver** process member names are shortened to **bkserver** and **rpserver**, respectively. For additional information, see the examples on page 488.

The following are the PDS member names for the DFS processes controlled by the **bosserver** process:

bkserver	Starts the Backup Server process. The name, **bkserver**, is an alias for the load library entry, **IOEBKSRV**. For further information, see "bakserver" on page 473.
flserver	Starts the Fileset Location Server process. The name, **flserver**, is an alias for the load library entry, **IOEFLSRV**. For further information, see "flserver" on page 605.
ftserver	Starts the Fileset Server process. The name, **ftserver**, is an alias for the load library entry, **IOEFTSRV**. For further information, see "ftserver" on page 720.
upclient	Starts the Update Client process. The name, **upclient**, is an alias for the load library entry, **IOEUPCLN**. For further information, see "upclient" on page 765.
upserver	Starts the Update Server process. The name, **upserver**, is an alias for the load library entry, **IOEUPSRV**. For further information, see "upserver" on page 768.
rpserver	Starts the Replication Server process. The name, **rpserver**, is an alias for the load library entry, **IOERPSRV**. For further information, see "repserver" on page 745.

For a **cron** process, provide two parameters. The first parameter is either the pathname to a binary file to be executed or the complete pathname of a command from one of the DFS suites (complete with all of the necessary arguments). Surround this parameter with "" (double quotes) if it contains spaces.

If the specified executable file does not exist, the **bos create** command does not create an entry in the **BosConfig** file. Instead, the command displays the following message:

```
bos:  unable to create a new server instance of
type process_type (specified executable not found)
```

The second parameter for a **cron** process specifies the time when the BOS server is to execute the command specified by the first parameter. Use a day and time together to execute the command weekly at the specified time; use a time alone to execute the command daily at the specified time. Day and time specifications have the following format:

[*day*]*hh*:*mm*

Enter the name of the day in all lowercase letters, giving either the whole name or the first three letters as an abbreviation (for example, **sunday** or **sun**). Specify the time of day by separating the hours from the minutes with a colon (:). Use 24-hour time, for example, **14:30**, or 1:00 to 12:00 with am or pm for example, **"2:30 pm"**. The time part of the option is optional if the day is specified; if time is excluded, it defaults to 00:00 on the specified day. As shown in the example, enclose the entire entry in double quotes if it contains spaces.

To execute the command only once, specify **now** instead of a day or a day and time, or issue the command directly; the process entry is removed from the **BosConfig** file after the command is executed. To place the process entry in the **BosConfig** file without ever executing it, specify **never** instead of a day or a day and time.

Important Note to Users

On OS/390 DFS, the **bos create -cmd** command option issues the **-path** command string for the **upclient** and **upserver** commands to start the **upclient** and **upserver** servers.

There is a limit of 256 characters that can be included in the command argument when using the **-cmd** option. If the 256 character limit is exceeded, unpredictable errors may result when the specified process starts.

To avoid this limitation when adding **upclient** or **upserver** processes on OS/390 DFS, requests for different files from the same directory may be combined. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The trailing slash (/) after the pathname must be included before the file name. The pathname and file names must be enclosed in parentheses (see the example on page 489 where **admin.up**, **admin.bos**, and **admin.ft** are specified and separated by spaces). For further information, see page 488, "upclient" on page 765, or "upserver" on page 768.

-noauth Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos create** command creates a new server process on the server machine specified by **-server** by creating an entry in the **BosConfig** file on the local disk of the machine. The status of the new process entry in both the **BosConfig** file and memory is set to **Run**, and the process is started on the server machine (unless the process is a **cron** process and the second parameter of the **-cmd** option is **never**).

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

In the following OS/390 DFS example, **/.:/hosts/DFSMVS** identifies the OS/390 host running the DFS server started task. The **ftserver simple** entry specifies the server process name and process type. The **ftserver** entry immediately following the first **"** (double quote) is the name of the member in the *xxx*.SIOELMOD data set (where *xxx* is installation dependent) in the OS/390 partitioned data set (PDS) load library (the *xxx*.SIOELMOD data set was created during installation). For further information, refer to the *OS/390 Program Directory*. The portion of the command in double quotes specifies the program, **ftserver**, to be executed, sets a **C** language runtime parameter, **envar**, and redirects **STDOUT** for this process to the OS/390 DDNAME **dd:ftserver**.

On OS/390 DFS, **envar**, the **C** runtime parameter, must include **_EUV_HOME** which points to that process' home directory. The slash, **/**, before **>dd:ftserver** separates the **C** runtime parameters from the OS/390 DFS program parameters. A redirection parameter to the server's own output DDNAME must be specified on OS/390 DFS. In this example, the redirection is specified by: **dd:ftserver** (the **2**>**&1** entry redirects **STDERR** to the same destination as **STDOUT**). It is important to note that OS/390 DFS may, depending on the processes to be added, require additional parameters.

```
$ bos create /.:/hosts/DFSMVS ftserver simple \
"ftserver envar('_EUV_HOME=/opt/dfslocal/home/ftserver')/ >dd:ftserver 2>&1"
```

The following example shows the syntax for adding a Fileset Location Server (**flserver**) on an OS/390 DFS system. The procedure is similar with the previous example.

```
$ bos create /.:/hosts/DCEFVT8 flserver simple \
"flserver envar('_EUV_HOME=/opt/dfslocal/home/flserver')/ >dd:flserver 2>&1"
```

The following example shows the syntax for adding a Backup Server (**bakserver**) on an OS/390 DFS system. Note the use of the process member name **bkserver**. All **simple** process commands on OS/390 DFS are members of the *xxx*.SIOELMOD (where *xxx* is installation dependent) data set created during installation. As all OS/390 member names are limited to eight or less characters, the **bakserver** process member name is shortened to **bkserver** in OS/390 DFS.

```
$ bos create /.:/hosts/DCEFVT8 bakserver simple \
"bkserver envar('_EUV_HOME=/opt/dfslocal/home/bakserver')/ >dd:bkserver 2>&1"
```

The following example shows the syntax for adding an update client under OS/390 DFS. In this example, the **upclient simple** entry specifies the server process name and process type. The **upclient** entry immediately following the first **"** (double quote) is the name of the module in the PDS. The portion of the command in double quotes specifies the program, **upclient**, to be executed, sets a **C** language runtime parameter, **envar**, and redirects **STDOUT** for this process to the OS/390 DDNAME **dd:upclient**.

The **C** runtime parameter, **envar**, includes **_EUV_HOME** which points to that process' home directory. The **C** runtime parameters are separated from the OS/390 DFS program parameters by a slash, **/**. In this example, the redirection is specified by **dd:upclient**.

On OS/390 DFS, additional parameters, such as administrative control lists, are entered after the redirection to **STDOUT**. In the example, parameters are entered identifying the path for server **wichita** and the files the client is to check. Note that the full pathname, **/opt/dcelocal/var/dfs/admin.up** is entered. In this example, the **admin.up**, **admin.bos**, and **admin.ft** files are specified. These are files containing administrative lists for the Update Server (**upserver**), the Basic OverSeer Server (**bosserver**) and Fileset Server (**ftserver**) processes. The **-f filename** parameter invokes a switch, **-f**, that points to a file, **filename**, to be used as a log file.

```
$ bos create /.:/hosts/DFSMVS upclient simple "upclient \
    envar('_EUV_HOME=/opt/dfslocal/home/upclient')/ >dd:upclient 2>&1 \
    /.:/hosts/wichita /opt/dcelocal/var/dfs/admin.up \
    /opt/dcelocal/var/dfs/admin.bos \
    /opt/dcelocal/var/dfs/admin.ft -f filename"
```

The following example shows another method for adding an update client on OS/390 DFS. In the following example, parameters are entered identifying the File Server machine **wichita** and the files the client is to check. The **-path** option specifies the the files to be checked. Note that the full pathname, **(/opt/dcelocal/var/dfs/ admin.up admin.bos admin.ft)** is entered followed by the requested files (separated by spaces). The pathname and file names are enclosed in parentheses. This will combine requests for different files from the same directory, significantly reducing the length of the command argument.

In this example, the **admin.up**, **admin.bos**, and **admin.ft** files are specified and separated by spaces. All reside in the same directory. These are files containing administrative lists for the Update Server (**upserver**), the Basic OverSeer Server (**bosserver**) and Fileset Server (**ftserver**) processes. The **-f filename** parameter invokes a switch, **-f**, that points to a file called **filename** that is to be used as a log file.

```
$ bos create /.:/hosts/DFSMVS upclient simple "upclient \
    envar('_EUV_HOME=/opt/dfslocal/home/upclient')/ >dd:upclient 2>&1 \
    -server /.:/hosts/wichita -path (/opt/dcelocal/var/dfs/ admin.up admin.bos admin.ft) \
    -f filename"
```

The following command creates the **simple** process **flserver** on the non-OS/390 machine named **aixfs3**:

```
$ bos create /.../abc.com/hosts/aixfs3 flserver simple /opt/dfsglobal/bin/flserver
```

The following is a sample **cron** process entry named **backup** on the machine named **aixfs1**. The **-localauth** option allows the unauthenticated process to use the DFS server principal of **aixfs1** to execute the privileged **fts clonesys** command.

```
$ bos create /.../abc.com/hosts/aixfs1 backup cron \
"IOEFTS clonesys -s /.../abc.com/hosts/aixfs1 -localauth" 5:30
```

For OS/390 DFS, the pathname for the program to be executed is not needed, instead the PDS member name is used. In the following example, the **fts clonesys** command is run at 5:30 am.

```
$ bos create /.:/hosts/mvsfs1 backup cron \
"IOEFTS clonesys -s /.:/hosts/mvsfs1 -localauth" 5:30
```

The following command creates the **simple** process, **repserver**, by executing the OS/390 program, **rpserver**. The **rpserver** entry immediately following the first **"** (double quote) is the name of the module in the PDS. The portion of the command in double quotes specifies the program, **rpserver**, to be executed, sets a **C** language runtime parameter, **envar**, and redirects **STDOUT** for this process to the OS/390 DDNAME **dd:rpserver**.

Note the use of the PDS member name, **rpserver**. As all OS/390 member names are limited to eight characters or less, the **repserver** process member name is shortened to **rpserver** on OS/390 DFS.

```
$ bos create /.:/hosts/DFSMVS repserver simple \
"rpserver envar('_EUV_HOME=/opt/dfslocal/home/repserver')/ >dd:rpserver 2>&1"
```

The following command creates the **simple** process, **upserver**, on an AIX machine, **DCEDFS7**. The portion of the command in ("") double quotes specifies the program, **upserver**, to be executed on the remote system.

```
$ bos create /.:/hosts/DCEDFS7 upserver simple              \
  "upserver envar('_EUV_HOME=/opt/dfslocal/home/upserver')/   \
  >dd:upserver 2>&1                                           \
  -path (/opt/dcelocal/var/dfs/ admin.up admin.bos admin.bak  \
  admin.ft admin.fl)"
```

Implementation Specifics

There are operating system-specific differences for issuing the **bos create** command against servers running under OS/390 DFS. These differences involve specifying a partitioned data set (PDS) member name entry in the **BosConfig**. In addition, the **bos create** command redirects output from servers running under the control of the **bosserver** to its own output OS/390 data definitions (DD). To accomplish this, the **bos create** command is modified slightly through a redirection parameter in the program parameter. This parameter must also include the definition of an environment variable to define the program's home directory. This allows the server running under OS/390 DFS to find the appropriate **krb5ccname** file and to initiate its own dce_login. The **path** for the program must be the partitioned data set (PDS) member name for the program to be run.

On OS/390 DFS, the **bos create -cmd** command option issues the **-path** command string for the **upclient** and **upserver** commands to start the **upclient** and **upserver** servers.

There is a limit of 256 characters that can be included in the command argument when using the **-cmd** option. If the 256 character limit is exceeded, unpredictable errors may result when the specified process starts.

To avoid this limitation when adding **upclient** or **upserver** processes on OS/390 DFS, requests for different files from the same directory may be combined. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The trailing slash (/) after the pathname must be included before the file name. The pathname and file names must be enclosed in parentheses (see the example on page 489 where **admin.up**, **admin.bos**, and **admin.ft** are specified and separated by spaces). For further information, see page 488, “upclient” on page 765, or “upserver” on page 768.

Related Information

Command:
bos delete
File:
BosConfig

bos delete

Purpose

Deletes server processes from the **BosConfig** file.

Format

bos delete -server *machine* **-process** *server_process*... **[{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine from which to delete one or more server processes. The BOS server on this machine executes the command. 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.

-process *server_process*
: Names each process to delete. Use the name assigned with the **-process** option in the **bos create** command; if necessary, use the **bos status** command to list the possible process names.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos delete** command removes each indicated server process entry from the **BosConfig** file on the server machine specified by **-server**. Before issuing this command, the issuer must use the **bos stop** command to stop each indicated process, both **simple** and **cron**, running on **-server**. An error message results if the status flag of a process being deleted is **Run** when this command is issued.

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following command removes the **ftserver** process entry from the **BosConfig** file on the machine named **fs3**:

```
$ bos delete /.../abc.com/hosts/fs3 ftserver
```

Related Information

Command:
bos create
File:
BosConfig

bos gckeys

Purpose

Removes obsolete server encryption keys from a keytab file.

Format

bos gckeys -server *machine* **[-principal** *name***] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine whose keytab file is to have obsolete keys removed from it. The BOS server on this machine executes the command. 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.

-principal *name*
: Provides the principal name for which obsolete keys are to be removed from the keytab file. The default is the DFS principal name of the machine specified by **-server**

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos gckeys** command removes obsolete server encryption keys from the **/krb5/v5srvtab** keytab file on the server machine specified by **-server**. Obsolete keys associated only with the principal name specified by **-principal** are removed from the keytab file; the DFS principal name of the server machine specified with **-server** is used by default.

Note: The **/krb5/v5srvtab** keytab file is a file included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

Keys are removed based on age and lack of use. The removal process, referred to as *garbage collection*, affects only the keytab file stored on the local disk of the machine indicated by **-server**; it has no effect on the Registry Database.

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**.

Output

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

Examples

The following command removes obsolete keys associated with the principal **hosts/fs1/dfs-server** from the keytab file on the server machine named **/.../abc.com/hosts/fs3**. Note that the keys being removed are associated with the principal name of a machine different from the one whose BOS server is executing the command.

```
$ bos gckeys /.../abc.com/hosts/fs3 hosts/fs1/dfs-server
```

Related Information

Commands:

bos addkey	**bos lskeys**	**dcecp**
bos genkey	**bos rmkey**	

File:
/krb5/v5srvtab

bos genkey

Purpose

Generates a random key and adds it to a keytab file.

Format

bos genkey -server *machine* **-kvno** *+_or_version_number* **[-principal** *name***] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
Names the server machine whose keytab file is to have a new key added to it. The BOS Server on this machine executes the command. 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.

-kvno *+_or_version_number*
Defines the key version number of the new key. The version number must be one of the following:

- An integer in the range 1 to 255. The command uses the specified number as the version number of the new key. The integer must be unique for the principal specified by **-principal** in the keytab file on the machine specified by **-server**.
- **+** or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer is unique for the principal indicated by **-principal** in the Registry database. However, it may not be unique for the indicated principal in the keytab file on the specified machine, in which case it replaces the key currently associated with the principal/version number pair in every keytab file.

The new key and its version number always replace the key and version number currently stored in the Registry database for the indicated principal.

-principal *name*
Provides the principal name with which the key is to be associated. The default is the DFS principal name of the machine specified by **-server**.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos genkey** command associates a new server encryption key with the principal name indicated by **-principal** in the **/krb5/v5srvtab** keytab file on the machine specified by **-server** and in the Registry Database. The command generates a random key and assigns it the version number indicated by **-kvno**. The issuer can either specify a version number or have the command choose one that is unique for the indicated principal in the Registry Database. The server encryption key and version number for the specified principal are automatically updated both in the keytab file on the specified server machine and in the Registry Database.

The **bos addkey** command can also be used to add a key to a keytab file with or without updating the Registry Database. However, it is less secure because the issuer must specify a string to be converted into the server encryption key. The keytab file must already exist before the **bos genkey** or **bos addkey** commands can be used to add a key to it. (Keytab files are created using the **dcecp** command.)

If the **bos rmkey** command is issued against the key that is currently being stored in the registry, the passwords get out of sequence (key's password in the local keytab file versus key's password in the registry). This happens because the **bos rmkey** command does not affect the registry as the **bos genkey** and **bos addkey** commands do when they are not used with the **-localonly** option. The following **rgy_edit** command can be used to get back into sequence:

```
rgy_edit> ktdelete -p <affected principal> -v <xx>

(do this for every xx version in the keytab file)

rgy_edit> do a
rgy_edit> change -p <affected principal> -g <group> -o <org -pw <newpw> -mp <ce
ll_admin's pw>
rgy_edit> ktadd -p <affected principal> -pw <newpw>
```

Note: The **/krb5/v5srvtab** keytab file and the **dcecp** command are included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

Privilege Required

You must be listed in the **admin.bos** file on the machine specified by **-server**, and the DFS server principal of the machine specified by **-server** must have the permissions necessary to alter entries in the Registry Database.

Output

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC. Continuing without it.
```

Examples

The following command generates a new server encryption key with key version number **14** and adds it to the keytab file on **fs1**. Because **-principal** is omitted, the key is associated with the DFS principal name of **fs1** (the machine specified with **-server**). The Registry Database is updated automatically.

```
$ bos genkey /.../abc.com/hosts/fs1 14
```

Related Information

Commands:

bos addkey	**bos lskeys**	**dcecp**
bos gckeys	**bos rmkey**	

File:
/krb5/v5srvtab

bos getdates

Purpose

Lists time stamps on versions of binary files.

Format

bos getdates -server *machine* **-file** *binary_file***...** **[-dir** *alternate_dest***] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine that houses the binary files whose time stamps are to be displayed. The BOS Server on this machine executes the command. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-file *binary_file*
: Names the current version of each binary file whose time stamps are to be displayed. The time stamps on the current, **.BAK**, and **.OLD** versions of each file are displayed. All specified files must reside in the same directory (**/opt/dfsglobal/bin**, by default, or an alternate directory specified with the **-dir** option). Specify only file names; if a pathname is provided for a file, the command ignores all but the final element.

-dir *alternate_dest*
: Provides the pathname of the directory in which all the specified files reside. Omit this option if the files reside in the default directory, **/opt/dfsglobal/bin**; otherwise provide a full or relative pathname. Relative pathnames (pathnames that do not begin with a slash) are interpreted relative to the **/opt/dfslocal** directory on the machine specified by **-server**.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as root for this option to work. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos getdates** command displays the time stamps and dates for the current, **.BAK**, and **.OLD** versions of each binary file whose current version is specified with the **-file** option. The time stamps record when the files were installed. The command displays a message for any version of a specified file that does not exist. Use the **-server** option to specify the name of the server machine where the files reside. The **-dir** option can be used to specify the name of the directory where the files reside if it is different from **/opt/dfsglobal/bin**.

The BOS Server automatically creates **.BAK** and **.OLD** versions when new binaries are installed with **bos install**. Use the **bos uninstall** command to replace the current version with its next oldest version (**.BAK** or, if the **.BAK** version does not exist, **.OLD**) or to remove all versions of a binary file. Use the **bos prune** command to remove **.BAK** and **.OLD** files from the **/opt/dfsglobal/bin** directory (the command can also be used to remove core files from the **/opt/dcelocal/var/dfs/adm** directory).

Privilege Required

No privileges are required.

Output

For each file specified with the **-file** option, the output reports the time stamp on the current, **.BAK**, and **.OLD** versions. The output displays a message to indicate any version that does not exist.

Examples

The following command displays the time stamps on the three versions of the **flserver** binary file stored in the default directory on the server machine named **fs2**:

```
$ bos getdates /.../abc.com/hosts/fs2 flserver
```

Implementation Specifics

The **bos getdates** command is not supported in OS/390 DFS when issued against the OS/390 **bosserver**. This command may be issued from OS/390 against other non-OS/390 **bosservers**.

Related Information

Commands:

bos install	**bos prune**	**bos uninstall**

bos getlog

Purpose

Examines the log file for a server or OS/390 client process.

Format

bos getlog -server *machine* **-file** *log_file* **[{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine from which to retrieve the log file. The BOS Server on this machine executes the command. 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.

-file *log_file*
: Names the log file to display. If a simple file name is provided, with no slashes, it is assumed to reside in **/opt/dcelocal/var/dfs/adm**; the standard choices from that directory are **BakLog**, **BosLog**, **CMLog**, **FlLog**, **FtLog**, **RepLog**, and **UpLog**.

: Pathnames are interpreted relative to **/opt/dcelocal/var/dfs/adm**; absolute pathnames are also allowed. In cases where a **/** (slash) appears in the specified file name, the issuer's username must appear in the **admin.bos** file on the machine specified by the **-server** option.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If the filename specified by **-file** contains a **/** (slash), the command is not successful if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

This command displays the contents of the log file specified by **-file** that resides on the machine specified by **-server**. It can be used to view any of the following log files:

BakLog	Generated by the Backup Server process on each backup database machine
BosLog	Generated by the BOS Server process on each server machine
CMLog	Generated by the Cache Manager on OS/390 client machine
FlLog	Generated by the Fileset Location Server process on each fileset database machine
FtLog	Generated by the Fileset Server process on each file server machine

RepLog	Generated by the Replication Server process on each server machine
UpLog	Generated by the **upserver** process on each server machine running the server portion of the Update Server

By default, the command looks in the **/opt/dcelocal/var/dfs/adm** directory for the log file it is to display. It is not necessary to specify the full pathname of a log file if it resides in the default directory. However, if the file resides elsewhere, the full pathname of the log file must be provided. (The command can also be used to view the **old** version of a log file created by the associated server process.)

Privilege Required

No privilege is required if the file name specified by **-file** does not contain a **/** (slash). If the name contains a **/** (slash), the issuer must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following example displays the contents of the **BosLog** file located in the default directory (**/opt/dcelocal/var/dfs/adm**) on the server machine named **fs1**:

```
$ bos getl /.../abc.com/hosts/fs1 BosLog
```

Related Information

Files:

BakLog	**FlLog**	**RepLog**
BosLog	**FtLog**	**UpLog**
CMLog		

bos getrestart

Purpose

Lists automatic restart times for server processes.

Format

bos getrestart -server *machine* **[{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine on which to check the restart times. The BOS Server on this machine executes the command. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos getrestart** command displays the following two restart times from the **BosConfig** file on the server machine specified by the **-server** option:

- The general restart time, which is the time each week when the BOS Server process automatically restarts itself and all processes that have the status flag of **Run** in the **BosConfig** file.
- The new binary restart time, which is the time each day when the BOS Server automatically restarts any process executed from a binary file in **/opt/dfsglobal/bin** whose time stamp is later than the last restart time for the process.

Either of these times can be daily (consist only of a time) or weekly (consist of a day and time). By default, the general restart time is once a week, while the new binary restart time occurs once a day. Both restart times are set with the **bos setrestart** command.

Privilege Required

No privileges are required.

Output

The output consists of the following two lines:

Server *machine* **restarts at** *time*

Server *machine* **restarts for new binaries at** *time*

where *machine* indicates the name of the server machine whose restart times are displayed, and possible values for *time* include

never Indicates that the BOS Server never performs that type of restart

A specified day and time
Indicates that the BOS Server performs that type of restart once per week

A specified time
Indicates that the BOS Server performs that type of restart once per day

Examples

The following command displays the restart times for the server machine **dcefvtl**:

```
$ bos getrestart -s dcefvt1

bos: WARNING: short name for server used; no authentication information
will be sent to the bosserver
Server dcefvt1 restarts at sun 4:00 am.
Server dcefvt1 restarts for new binaries never.
```

Implementation Specifics

The new binary restart time is not meaningful in OS/390 DFS when displayed from an OS/390 **bosserver**. This information may be requested from OS/390 from a non-OS/390 **bosserver**.

Related Information

Command:
bos setrestart
File:
BosConfig

bos help

Purpose

Shows syntax of specified **bos** commands or list functional descriptions of all **bos** commands.

Format

bos help [-topic *string*...**] [-help]**

Options

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **status**, not **bos status**). If this option is omitted, the output provides a short description of all **bos** commands.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos help** command displays the first line (name and short description) of the online help entry for every **bos** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **bos apropos** command to show each help entry containing a specified string.

Output

The online help entry for each **bos** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Privilege Required

No privileges are required.

Examples

The following command displays the online help entry for the **bos status** command:

```
$ bos help status

bos status: show server process status
Usage: bos status -server <machine> [-process server_process>...] [-long] \
[-noauth] [-localauth] [-help]
```

Related Information

Command:
bos apropos

bos install

Purpose

Installs new versions of a binary file.

Usage

The **bos install** command installs each binary file specified with the **-file** option on the server machine specified with the **-server** option. The **-file** option provides the pathname of each file to be installed on the specified machine. By default, the command installs the files in the **/opt/dfsglobal/bin** directory on the specified machine; use the **-dir** option to indicate a different directory on the specified machine.

Important Note to Users

The **bos install** command is not supported in OS/390 DFS.

bos lsadmin

Purpose

Lists the users, groups, and servers from an administrative list.

Format

bos lsadmin -server *machine* **-adminlist** *filename* **[{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine that houses the administrative list whose principals and groups are to be displayed. The BOS server on this machine executes the command. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-adminlist *filename*
: Names the administrative list whose principals and groups are to be displayed. The complete pathname is unnecessary if the list is stored in the default configuration directory (**/opt/dcelocal/var/dfs**).

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. In OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos lsadmin** command lists the principal names of users and server machines and the names of groups found in the administrative list specified by the **-adminlist** option on the server machine specified by the **-server** option. Principals whose names are specified in the administrative list or that are members of groups specified in the list can issue administrative commands for the DFS server process associated with the list.

The default path for the administrative lists is the configuration directory (**/opt/dcelocal/var/dfs**). If the specified list is stored in the default directory, only the specific file name is required. If the specified list is stored elsewhere, the pathname to the file that was used when the associated server process was started is required.

Use the **bos addadmin** command to add principals and groups to an administrative list. Use the **bos rmadmin** command to remove principals and groups from an administrative list.

Privilege Required

No privileges are required.

Output

The **bos lsadmin** command displays the output

```
Admin Users are:
```

followed by the principal name of each user and machine and the name of each group contained in the administrative list. Names from the local cell are displayed in an abbreviated form (for example, *username* for /.../*cellname*/*username*); names from foreign cells are displayed in full. Each name is preceded by one of the following strings:

`user:`	Precedes the principal name of each user or machine from the local cell
`foreign_user:`	Precedes the principal name of each user or machine from a foreign cell
`group:`	Precedes the name of each group from the local cell
`foreign_group:`	Precedes the name of each group from a foreign cell

Examples

The following command lists the members of the **admin.bos** file on the server machine named **fs1**. The administrative list contains two users, a server machine, and two groups, all of which are from the local cell.

```
$ bos lsa -server /.../abc.com/hosts/fs1 -adminlist admin.bos

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

Related Information

Commands:

bos addadmin	**bos rmadmin**	

Files:

admin.bak	**admin.fl**	**admin.up**
admin.bos	**admin.ft**	

bos lscell

Purpose

Lists the cell in which the BOS server is running

Format

bos lscell -server *machine* **[{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine on which the BOS server whose cell is to be listed is running. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. In OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos lscell** command reports the name of the cell in which the BOS server on the machine specified with the **-server** option is running. The command extracts information from the local configuration file, **/opt/dfslocal/dce_cf.db**, on the specified machine. If the command fails after being issued from the machine specified by **-server** (if **-server** is the local machine), the failure may indicate that the local **dce_cf.db** file is corrupted; use the **cat** or **more** command (or a similar command appropriate to your operating system) to display the contents of the file, and ensure that it is not corrupted.

Privilege Required

No privileges are required.

Output

This command displays the following line reporting the name of the cell in which the BOS server is running:

Cell name is *cellname*

Examples

The following command displays the name of the cell in which the BOS server on the machine named **fs1** is running:

```
$ bos lscell /.../abc.com/hosts/fs1

Cell name is abc.com
```

bos lskeys

Purpose

Displays server encryption key information from a keytab file.

Format

bos lskeys -server *machine* **[-principal** *name***] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine whose keytab file is to have keys listed. The BOS server on this machine executes the command. 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.

-principal *name*
: Provides the principal name for which associated keys are to be listed. The default is the DFS principal name of the machine specified by **-server**.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). The issuer must be logged in to the machine as **root** for this option to work. In OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos lskeys** command formats and displays information about server encryption keys kept in the **/krb5/v5srvtab** keytab file on the server machine specified by **-server**. It displays information for keys associated with the principal name indicated by **-principal**; the DFS principal name of the server machine specified with **-server** is used by default.

Note: The **/krb5/v5srvtab** keytab file is a file included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

The DFS authorization checking must be disabled on the machine specified by **-server** to display the string of octal numbers that compose the key (use the **bos setauth** command to disable DFS authorization checking). Disabling DFS authorization checking is required for two reasons. First, it implies that only someone authorized to issue the **bos setauth** command or someone with **root** access to **-server's** local disk (presumably a system administrator) is able to see actual encryption keys. Second, it makes it clear that the system is in a compromised state of security while server encryption keys are being examined (both turning off authorization checking and displaying keys on a screen are serious security risks).

If DFS authorization checking is enabled on **-server** (the normal case), a checksum appears in place of the octal numbers. A *checksum* is a decimal number derived by encrypting a constant with each key.

Privilege Required

If DFS authorization checking is enabled, you must be listed in the **admin.bos** file in the machine specified by **-server**; checksums are displayed instead of the actual keys. Because DFS authorization checking must be disabled with the **bos setauth** command before the actual keys (rather than just checksums) can be displayed, no privilege is required to see the keys. However, you must be listed in the **admin.bos** file on a machine to use the **bos setauth** command to disable DFS authorization checking on it.

Output

The **bos lskeys** command displays one line for each server encryption key associated with **-principal** in the keytab file on the machine specified by **-server**. Each key is identified by its key version number. If DFS authorization checking is enabled on the machine, a checksum is displayed with each version number; if authorization checking is disabled, the octal numbers that comprise the key are displayed.

A line specifying when the key in the Registry Database (at the Registry Server) was last changed follows the list of keys or checksums. The words **All done** indicate the end of the output.

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC.  Continuing without it.
```

Examples

The following command shows the checksums for the keys associated with the principal name of **fs3** in the keytab file on that machine. The checksums appear instead of the actual keys because DFS authorization checking is *not* disabled.

```
$ bos lsk /.../abc.com/hosts/fs3

key 1 has cksum 972037177
key 3 has cksum 282517022
key 4 has cksum 260617746
Keys last changed (at the registry server) on Thu Jun 6 11:24:46 1991.
All done.
```

The following command lists the keys associated with **fs3** after DFS authorization checking is disabled with **bos setauth** command:

```
$ bos setauth /.../abc.com/hosts/fs3 off
$ bos lsk /.../abc.com/hosts/fs3

key 1 is '\040\205\211\241\345\002\023\211'
key 2 is '\343\315\307\227\255\320\135\244'
key 3 is '\310\310\255\253\265\236\261\211'
Keys last changed (at the registry server) on Thu Jun 6 11:24:46 1991.
All done.
```

Related Information

Commands:

bos addkey	**bos genkey**	**bos setauth**
bos gckeys	**bos rmkey**	**dcecp**

File:
/krb5/v5srvtab

bos prune

Purpose

Removes old binary and core files from **/opt/dfsglobal/bin** and **/opt/dcelocal/var/dfs/adm**.

Usage

The **bos prune** command removes obsolete versions of binary and core files from the **/opt/dfsglobal/bin** and **/opt/dcelocal/var/dfs/adm** directories on the server machine specified with the **-server** option. Binary files should only need to be removed from the Binary Distribution machine for a CPU/operating system type; core files may need to be removed from any server machine.

Important Note to Users

The **bos prune** command is not supported in OS/390 DFS.

bos restart

Purpose

Restarts processes on a server machine.

Format

bos restart -server *machine* **[{-bosserver | -process** *server_process*...**}] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine on which to stop and restart the indicated processes. 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.

-bosserver
: Indicates that all processes, including the current BOS server, are to stop running. A new BOS server immediately starts; it then restarts all processes with the status flag **Run** in the **/opt/dcelocal/var/dfs/BosConfig** file.

  Provide this option or provide the **-process** option. Omit both options to stop all processes except the BOS server; those with the status flag **Run** in the **BosConfig** file are immediately restarted.

-process *server_process*
: Specifies each process to be stopped and immediately restarted. The BOS server stops all specified processes that are currently running; it then restarts all of the specified processes, regardless of their status flags in the **BosConfig** file. Refer to a process by the name assigned with the **-process** option of the **bos create** command (this name appears in the output from the **bos status** command). *Do not include* **bosserver** *in the list of processes*; use the **-bosserver** option instead.

  Provide this option or provide the **-bosserver** option. Omit both options to stop all processes except the BOS server; those with the status flag **Run** in the **BosConfig** file are immediately restarted.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. In OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos restart** command instructs the BOS server running on the server machine specified by **-server** to stop all indicated processes on the machine. The BOS server then immediately restarts some or all of the processes, depending on the options included with the command. The processes to be stopped and possibly restarted are specified with the following options:

- The **-bosserver** option causes the BOS server to stop all processes including itself. A new BOS server immediately starts; it then restarts all processes with the status flag **Run** in the **BosConfig** file.
- The **-process** option causes the BOS server to stop and immediately restart all specified processes, regardless of their status flags in the **BosConfig** file. All restarted processes with the status flag **NotRun** in the **BosConfig** file have the status **temporarily enabled** in the output of the **bos status** command.
- The absence of both the **-bosserver** and **-process** options causes the BOS server to stop all processes except itself. The BOS server then immediately restarts all processes with the status flag **Run** in the **BosConfig** file.

This command can be used to stop only those processes the BOS server controls. Also, it does *not* change the status flag for a process in the **BosConfig** file.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following command instructs the BOS server on **/.../abc.com/hosts/fs3** to stop all processes, including itself. A new BOS server immediately starts, after which it restarts all processes with the status flag **Run** in the **BosConfig** file.

```
$ bos res /.../abc.com/hosts/fs3 -bos
```

The following command instructs the BOS server on **/abc.com/hosts/fs5** to stop all processes except itself. The BOS server then restarts all processes marked with the status flag **Run** in the **BosConfig** file.

```
$ bos restart /.../abc.com/hosts/fs5
```

Related Information

Commands:

bos create	**bos status**

File:
BosConfig

bos rmadmin

Purpose

Removes a user, group, or server from an administrative list.

Format

bos rmadmin -server *machine* **-adminlist** *filename* **[-principal** *name***...] [-group** *name***...] [-removelist] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
Names the server machine that houses the administrative list from which principals, groups, or both are to be removed. The BOS server on this machine executes the command. 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.

-adminlist *filename*
Names the administrative list from which principals, groups, or both are to be removed. The complete pathname is unnecessary if the list is stored in the default configuration directory (**/opt/dcelocal/var/dfs**).

-principal *name*
Specifies the principal name of each user or server machine to be removed from the administrative list. A user from the local cell can be specified by a full or abbreviated principal name (for example, /**...**/*cellname*/*username* or just *username*); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, /**...**/*cellname*/**hosts**/*hostname*/**self** or just **hosts**/*hostname*/**self**); a server machine from a foreign cell can be specified only by a full principal name.

-group *name*
Specifies the name of each group to be removed from the administrative list. A group from the local cell can be specified by a full or abbreviated group name (for example, /**...**/*cellname*/*group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

-removelist
Specifies that the file indicated with **-adminlist** is to be removed if it is empty either when the command is issued or after any principals or groups specified with the command are removed. This option has no effect if the specified file is not empty when the command is issued or after any indicated principals or groups are removed.

-noauth Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database).

You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos rmadmin** command removes the specified users, groups, and servers from the administrative list specified by the **-adminlist** option on the server machine specified by the **-server** option. The principal (login) names of users and the principal names of server machines to be removed from the administrative list are specified with the **-principal** option; the names of groups to be removed from the list are specified with the **-group** option. Principals removed from the administrative list either directly (with the **-principal** option) or indirectly (as members of groups indicated with the **-group** option) can no longer issue administrative commands for the DFS server process associated with the list.

The default path for administrative lists is the configuration directory (**/opt/dcelocal/var/dfs**). If the specified list is stored in the default directory, only the specific file name is required. If the specified list is stored elsewhere, the pathname to the file that was used when the associated server process was started is required.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following command removes the former administrative users **smith** and **jones** from the **admin.bos** file on **fs1**:

```
$ bos rmadmin -server /.../abc.com/hosts/fs1 -adminlist admin.bos -principal smith jones
```

Related Information

Commands:

bos addadmin	**bos lsadmin**

Files:

admin.bak	**admin.fl**	**admin.up**
admin.bos	**admin.ft**	

bos rmkey

Purpose

Removes server encryption keys from a keytab file.

Format

bos rmkey -server *machine* **-kvno** *version_number*.... **[-principal** *name*] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine whose keytab file is to have keys removed from it. The BOS server on this machine executes the command. 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.

-kvno *version_number*
: Specifies the key version number of each key to be removed from the keytab file. The command removes each key that is associated with a specified key version number and the principal indicated by **-principal**. Each version number must be an integer in the range 1 to 255.

-principal *name*
: Provides the principal name associated with the keys to be removed from the keytab file. The default is the DFS principal name of the machine specified by **-server**

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos rmkey** command removes server encryption keys from the **/krb5/v5srvtab** keytab file on the server machine specified by **-server**. It removes each key associated with a key version number indicated by **-kvno** and the principal indicated by **-principal**. The command has no effect on the Registry Database. Once a key is removed from the keytab file, it can no longer be used to establish communication between clients and the server to which it applied.

If the **bos rmkey** command is issued against the key that is currently being stored in the registry, the passwords then get out of sequence (key's password in the local keytab file versus key's password in the registry). This happens because the **bos rmkey** command does not affect the registry as the **bos genkey** and **bos addkey** commands do when they are not used with the **-localonly** option. The following **rgy_edit** command can be used to get back into sequence:

```
rgy_edit> ktdelete -p <affected principal> -v <xx>

(do this for every xx version in the keytab file)

rgy_edit> do a
rgy_edit> change -p <affected principal> -g <group> -o <org -pw <newpw> -mp <ce
ll_admin's pw>
rgy_edit> ktadd -p <affected principal> -pw <newpw>
```

Note: The **/krb5/v5srvtab** keytab file is a file included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Output

If the packet privacy protection level is not available to you, the command displays the following message reporting that the BOS server is using the packet integrity protection level instead:

```
Data encryption unsupported by RPC.  Continuing without it.
```

Examples

The following command removes two keys from the keytab file on **fs1**: the keys with key version numbers **5** and **6** that are associated with the DFS principal name of **fs1**.

```
$ bos rmk /.../abc.com/hosts/fs1 -kvno 5 6
```

Related Information

Commands:

bos addkey	**bos genkey**	**dcecp**
bos gckeys	**bos lskeys**	

File:
/krb5/v5srvtab

Note: The **dcecp** command is included as part of the OS/390 DCE (OS/390 DCE Base). For further information, refer to *OS/390 DCE Command Reference* and *OS/390 DCE Administration Guide*.

bos setauth

Purpose

Enables or disables DFS authorization checking for all DFS server processes on a machine.

Format

bos setauth -server *machine* **-authchecking {on | off} [{-noauth | -localauth}] [-help]**

Options

-server *machine*
Names the server machine on which the status of DFS authorization checking is to change. The BOS Server on this machine executes the command. 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.

-authchecking
Determines whether or not server processes on the machine check for DFS authorization. A value of **on** enables DFS authorization checking; a value of **off** disables it.

-noauth
Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server** (the option can be used only when enabling authorization checking). If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **noauth** option.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos setauth** command enables or disables DFS authorization checking on the server machine specified by the **-server** option. If DFS authorization checking is enabled on a server machine, all DFS server processes running on the machine check that the issuer of a command is correctly authorized (is included in the necessary administrative lists) to execute the command. If DFS authorization checking is disabled on a server machine, the DFS server processes on the machine perform any action for any user, even the unprivileged user **nobody**.

By default, DFS authorization checking is enabled on every server machine. Disabling it on a server machine is a serious security risk. It is typically disabled for the briefest possible time and only in the following situations:

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file.

To indicate to all DFS server processes (including itself) that DFS authorization checking is disabled on a server machine, the BOS Server creates the zero length file **/opt/dcelocal/var/dfs/NoAuth** on the local disk of the machine. All DFS server processes, including the BOS Server, check for the presence of this file when they are requested to perform an operation; they do not check for the necessary administrative privilege for a requested operation when the file is present. To indicate that DFS authorization checking is enabled, the BOS Server removes the file.

Enter this command with the **-authchecking** option and an argument of **off** to disable DFS authorization checking on a server machine. (DFS authorization checking can also be disabled by including the **-noauth** option with the **bosserver** command used to start the BOS Server.) Issue the command with the **-authchecking** option and an argument of **on** to enable DFS authorization checking on a server machine. It is not necessary to restart currently running server processes when you change the state of DFS authorization checking; server processes immediately obey the current state of DFS authorization checking and act accordingly.

The **bos status** command can be used to determine whether DFS authorization checking is enabled or disabled on a server machine. The command displays the following message if DFS authorization checking is disabled on a machine (it does not display the message if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

The **-noauth** option available with many **bos** and **fts** commands is used when authentication information is unnecessary or unavailable. Use the **-noauth** option if DFS authorization checking is disabled on a server machine on which administrative privilege is required or if the Security Service is unavailable.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server** to disable DFS authorization checking on that machine. (No privilege is required to enable DFS authorization checking if it is currently disabled.)

Cautions

Always use the **bos setauth** command to create the **/opt/dcelocal/var/dfs/NoAuth** file. Do not create the file directly except when explicitly told to do so by instructions for dealing with emergencies (such as emergencies involving server encryption keys). Creating the file directly requires logging into the local operating system of a machine as **root** and using the **touch** command (or its equivalent). On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Examples

The following command disables DFS authorization checking for all DFS server processes on the server machine named **fs7**:

```
$ bos seta /.../abc.com/hosts/fs7 off
```

Related Information

Commands:

bos status	**bosserver**

File:

NoAuth

bos setrestart

Purpose

Sets the date and time at which the BOS Server restarts all processes or only those with new binaries.

Format

bos setrestart -server *machine* **{-general** *time* **| -newbinary** *time***} [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Specifies the server machine for which restart times are to be set. The BOS Server on this machine executes the command. 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.

-general *time*
: Sets the time when the BOS Server restarts first itself then restarts each server process that has an entry in the **BosConfig** file with a status flag of **Run**. Specify a day and time to perform the restart weekly at that time; specify a time to perform the restart daily at that time. Day and time specifications have the following format:

  [*day*]*hh*:*mm*

  Enter the name of the day in all lowercase letters, giving either the whole name or the first three letters as an abbreviation (for example, sunday or sun). Specify the time of day by separating the hours from the minutes with a colon. Use 24-hour time (for example, 14:30), or 1:00 through 12:00 with am or pm (for example, "12:30 pm"). As shown in the example, enclose the entry in quotes if it contains a space.

  Also, the issuer can use either of two additional definitions instead of a day and time:

  never Directs the BOS Server never to perform the indicated type of restart

  now Directs the BOS Server to use the day and time at which the command is issued (for example, Sunday at 2:00 a.m.) as the day and time for the indicated restart

  If this option is never used to set the general restart time in OS/390 DFS, the default general restart time is never. If this option is never used to set the general restart time, in non-OS/390 DFS systems, the default general restart time is Sunday at 4:00 a.m.

-newbinary *time*
: Sets the time at which the BOS Server restarts any server process whose binary file was installed in **/opt/dfsglobal/bin** after the current instance of the process started running. The recommended frequency is once per day, so it is standard to specify only a time of day. Use the conventions described for times under the **-general** option to express the time of day, and enclose it in double quotes if it contains a space. The remarks under the **-general** option concerning **never** and **now** also apply to this option.

  If this option is never used to set the binary checking time, the default binary checking time is 5:00 a.m.

Important Note to Users

The **-newbinary** option applies only to binary files on non-OS/390 platforms.

-noauth Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos setrestart** command sets the times at which the BOS Server running on the server machine specified by **-server** is to perform one of two types of restarts. The **bos setrestart** command records the time settings in the **BosConfig** file. The two types of restart times are:

- The time each week when the BOS Server restarts itself and any processes marked with the status flag **Run** in the **BosConfig** file. This is equivalent to executing **bos restart** with the **-bosserver** option.
- The time each day when the BOS Server restarts any process currently running for which the binary file in **/opt/dfsglobal/bin** was modified since the process was last started (or restarted). The default is 5:00 a.m. each day.

You must issue the command twice to change both times.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by the **-server** option.

Cautions

Restarting processes makes them unavailable for a period of time. It is advisable to set the restarts for times of typically low usage to inconvenience as few users as possible.

If the specified time is within one hour of the current time, the BOS Server does not restart the processes until that time on the next day.

Examples

The following command defines a general restart time in the **BosConfig** file on **fs4** that causes all processes on that machine to stop and restart each Saturday morning at 3:30 a.m.:

```
$ bos setr -s /.../abc.com/hosts/fs4 -gen "sat 3:30"
```

The following command defines a restart time in the **BosConfig** file on **fs6**, instructing the BOS Server on that machine to check for new binary files each evening at 11:45 p.m. and restart any processes for which it finds a new file at that time:

```
$ bos setr -s /.../abc.com/hosts/fs6 -new 23:45
```

Implementation Specifics

The **-newbinary** option applies only to binary files on non-OS/390 platforms.

If this option is never used to set the general restart time in OS/390 DFS, the default general restart time is never. If this option is never used to set the general restart time, in non-OS/390 DFS systems, the default general restart time is Sunday at 4:00 a.m.

Related Information

Commands:

bos getrestart	**bos restart**

File:
BosConfig

bos shutdown

Purpose

Stops processes without changing their status flags in the **BosConfig** file.

Format

bos shutdown -server *machine* **[-process** *server_process***...] [-wait] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine on which the indicated processes are to be stopped. The BOS Server on this machine executes the command. 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.

-process *server_process*
: Specifies each process to be stopped. If this option is omitted, the BOS Server stops all server processes other than itself on the server machine. Refer to a process by the name assigned with the **-process** option of the **bos create** command; this name appears in the output of the **bos status** command.

-wait
: Indicates that the command shell prompt is not to return until the shutdown is complete (until all processes actually stop running). If this option is omitted, the prompt returns almost immediately, even if all of the processes are not yet stopped.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos shutdown** command instructs the BOS Server running on the server machine specified by **-server** to stop either all processes (except itself) running on the machine *or* only the processes specified by **-process**. The command does not change a process' status flag in the **BosConfig** file, only in the BOS Server's memory state.

Processes stopped with this command do not run again until they are started using the **bos start**, **bos startup**, or **bos restart** commands, or until the BOS Server itself restarts.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following command instructs the BOS Server running on **fs3** to stop running all processes except itself:

```
$ bos shutdown -s /.../abc.com/hosts/fs3
```

The following command instructs the BOS Server running on **fs3** to stop running the **repserver** process:

```
$ bos shutdown -s /.../abc.com/hosts/fs3 -p repserver
```

Implementation Specifics

On OS/390 DFS, the **bos shutdown** command stops all processes controlled by the **bosserver**. These include the: **bakserver**, **flserver**, **ftserver**, **repserver**, **upserver**, and **upclient** processes. The remaining OS/390 DFS processes are controlled by the OS/390 system command **MODIFY**. For further information, see Chapter 17, "OS/390 System Commands" on page 327.

Related Information

Commands:

bos create	**bos status**

bos start

Purpose

Starts processes after setting their status flags to **Run** in the **BosConfig** file and in memory.

Format

bos start -server *machine* **-process** *server_process...* **[{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine where each process is to be started. 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.

-process *server_process*
: Specifies each process to be started after its status flag in the **BosConfig** file and in memory is set to **Run**. Refer to a process by the name assigned with the **-process** option of the **bos create** command; this name appears in the output from the **bos status** command.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command is not successful if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos start** command changes the status flag for each server process specified by **-process** from **NotRun** to **Run** in the **BosConfig** file and in memory on the server machine specified by **-server**. It then starts each specified process running on that machine.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by the **-server** option.

Cautions

If an instance of a process is already running, the only effect is to guarantee that its status flag is set to **Run** in both the **BosConfig** file and memory; it does not start a new instance of the process. Issue the **bos restart** command after this command to start a new instance.

Examples

The following command causes the BOS Server on **fs3** to start the Replication Server (**repserver** process) on that machine by changing its status flags to **Run** in both the **BosConfig** file and memory:

```
$ bos start /.../abc.com/hosts/fs3 repserver
```

Implementation Specifics

On OS/390 DFS, the **bos start** command starts all processes controlled by the **bosserver**. These include the: **bakserver**, **flserver**, **ftserver**, **repserver**, **upserver**, and **upclient** processes. The remaining OS/390 DFS processes are controlled by the OS/390 system command **MODIFY**. For further information, see Chapter 17, "OS/390 System Commands" on page 327.

Related Information

Commands:

bos create	**bos startup**	**bos status**
bos restart		

File:
BosConfig

bos startup

Purpose

Starts processes by changing their status flags to **Run** in memory without changing their status flags in the **BosConfig** file.

Format

bos startup -server *machine* **[-process** *server_process...***] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine whose processes are to be started. The BOS Server on this machine executes the command. 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.

-process *server_process*
: Specifies each process to be started after its status flag in memory is set to **Run**. Refer to a process by the name assigned with the **-process** option of the **bos create** command; this name appears in the output from the **bos status** command.

  If this option is omitted, all server processes with a status flag of **Run** in the **BosConfig** file that are not running are started after their status flags in memory are set to **Run**.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in as **root** to the machine for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

This command instructs the BOS Server running on the server machine specified by **-server** to start *either* all server processes with a status flag of **Run** in the **BosConfig** file that are not running (if **-process** is omitted) *or* each process specified by **-process**, even if its status flag in the **BosConfig** file is **NotRun**. The status flags of all started processes are changed from **NotRun** to **Run** in memory.

Using **-process** is useful for testing server processes without enabling them permanently. This command does **not** change the status flag for a process in the **BosConfig** file.

Cautions

If an instance of a process is already running, the only effect is to guarantee that its status flag is set to **Run** in memory; it does not start a new instance of the process. Issue the **bos restart** command after this command to start a new instance.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by **-server**.

Examples

The following command causes the BOS Server on **fs3** to start all processes on that machine marked with a status flag of **Run** in the **BosConfig** file that are not currently running. The status flags of all such processes are set to **Run** in memory; their status flags remain set to **Run** in the **BosConfig** file.

```
$ bos startup /.../abc.com/hosts/fs3
```

The following command causes the BOS Server on **fs3** to start the Replication Server (**repserver** process) on that machine by changing its status flag to **Run** in memory. The process' status flag remains unchanged in the **BosConfig** file, regardless of its current setting (**Run** or **NotRun**).

```
$ bos startup /.../abc.com/hosts/fs3 repserver
```

Implementation Specifics

On OS/390 DFS, the **bos startup** command starts processes controlled by the **bosserver**. These include the: **bakserver**, **flserver**, **ftserver**, **repserver**, **upserver**, and **upclient** processes. The remaining OS/390 DFS processes are controlled by the OS/390 system command **MODIFY**. For further information, see “modify dfs daemon” on page 329.

Related Information

Commands:

bos create	**bos start**	**bos shutdown**
bos restart	**bos status**	**bos stop**

File:
BosConfig

bos status

Purpose

Displays statuses of server processes on a server machine.

Format

bos status -server *machine* **[-process** *server process***...] [-long] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine about whose processes status information is to be displayed. The BOS Server on this machine executes the command. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-process *server_process*
: Specifies each process whose status is to be displayed; refer to a process by the name assigned with the **-process** option of the **bos create** command. If this option is omitted, the statuses of all of the processes on the specified server are listed.

-long
: Directs the BOS Server to provide more detailed information about the specified processes.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). On OS/390 DFS, you must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos status** command lists status information about the processes on the server machine specified by the **-server** option. Use the **-process** option to indicate the specific processes about which information is to be displayed, or omit the option to display information about all the processes on the server machine. The command also displays appropriate messages if DFS authorization checking is disabled on the machine or if the machine's **/opt/dfslocal** directory or a directory or file beneath it has inappropriate protections.

Use the **-long** option to display more information about each specified process. The additional information can be used to determine the role of a server machine in a domain. Refer to Chapter 9, “Monitoring and Controlling Server Processes” on page 129 for instructions on using this command to determine the role of a server machine.

Privilege Required

No privileges are required.

Output

The command first displays the following line if DFS authorization checking is disabled on the machine (the line is not displayed 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 **/opt/dfslocal** directory or a directory of file under it on the machine has protections that it believes are inappropriate:

```
Bosserver reports inappropriate access on server directories.
```

The message usually indicates that users who are not able to write to the **/opt/dfslocal** directory and its subdirectories have write access. The BOS server displays the message if the UNIX permission bits on the following objects do not enforce the indicated protections. Provided the mode bits do not violate the specific restrictions cited in the list, a directory or file can grant more permissions than those shown in the list, but it should not grant fewer.

/opt/dfslocal
At least **755**, and **other** cannot have write access

/opt/dfsglobal/bin
At least **755**, and **other** cannot have write access

/opt/dfslocal/var
At least **755**, and **other** cannot have write access

/opt/dcelocal/var/dfs
At least **701**, and **other** cannot have write access

/opt/dcelocal/var/dfs/adm
At least **755**, and **other** cannot have write access

/opt/dcelocal/var/dfs/admin.bos
At least **600**, and **other** cannot have write or execute access

The BOS Server also displays the message if all of these objects are not owned by **root**. On OS/390 DFS, **root** refers to a user with a **UID = 0**. The BOS server displays the message only as a courtesy to the user. It does nothing to change the protections on these objects, nor does it fail if these protections are violated.

The command then displays a separate entry for each specified process. The first line of an entry shows the current status of the process. The possible statuses for any process include

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 **/opt/dcelocal/var/dfs/BosConfig** file is **NotRun**, but the process has been enabled with the **bos startup** or **bos restart** command.

temporarily disabled
Either the **bos shutdown** command was used to stop the process, or the BOS Server quit trying to restart the process, in which case the message

```
stopped for too many errors
```

also appears.

disabled The status flag for the process in the **BosConfig** file is **NotRun**, and the process has not been enabled.

has core file
The process failed or produced a core file at some time. This message can appear with any of the other messages. Core files are stored in **/opt/dcelocal/var/dfs/adm**. The name of the core file indicates the process that failed (for example, **core.ftserver**).

currently shutdown
The status flag for the process in the **BosConfig** file is **NotRun**, and the process is not currently running.

The output for a **cron** process includes an auxiliary status message that reports when the command is next scheduled to execute.

The command displays the following additional information is displayed when the **-long** option is used:

- The process type (**simple** or **cron**).
- How many **proc starts** occurred (**proc starts** occur when the process is started or restarted by the current BOS Server).
- The time of the last **proc start**.
- The exit time and error exit time when the process last failed. This appears only if the process failed while the BOS Server was running. (Provided the BOS Server was running both when the process was started and when it failed, the BOS Server can provide this information for any process that has an entry in the **BosConfig** file.)
- The command and its options used by the BOS Server to start the process.

Example

The following command displays the statuses of all server processes on the File Server machine named **fs4**:

```
$ bos status /.../abc.com/hosts/fs4
Instance ftserver, currently running normally.
Instance repserver, currently running normally.
```

If the **-long** option is included with the command, the following additional information is displayed:

```
Instance ftserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:02 1991 (1 proc starts)
Parameter 1 is ‘/opt/dfsglobal/bin/ftserver’

Instance repserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:48 1991 (1 proc starts)
Parameter 1 is ‘/opt/dfsglobal/bin/repserver’
```

Implementation Specifics

On OS/390 DFS, the **bos status** command displays the status of server processes on server machines controlled by the **bosserver**. These include the: **bakserver**, **flserver**, **ftserver**, **repserver**, **upserver**, and **upclient** processes. The remaining OS/390 DFS processes are controlled by the OS/390 system command **MODIFY**. For further information, see “modify dfs daemon” on page 329.

Related Information

Commands:

bos create	**bos shutdown**	**bos startup**
bos restart	**bos start**	**bos stop**

File:

BosConfig

bos stop

Purpose

Stops processes after changing their status flags in the **BosConfig** file to **NotRun**.

Format

bos stop -server *machine* **-process** *server_process*... **[-wait] [{-noauth | -localauth}] [-help]**

Options

-server *machine*
: Names the server machine on which to stop the processes. The BOS Server on this machine executes the command. 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.

-process *server_process*
: Specifies each process the BOS Server is to stop. The BOS Server stops a process after setting its status flag in the **BosConfig** file to **NotRun**. Refer to a process by the name assigned with the **-process** option of the **bos create** command; this name appears in the output from the **bos status** command.

-wait
: Indicates that the command shell prompt is not to return until all specified processes actually stop running. If this option is omitted, the prompt returns almost immediately, even if all of the processes are not yet stopped.

-noauth
: Directs **bos** to use the unprivileged identity **nobody** as the identity of the issuer of the command. The command fails if you use this option and DFS authorization checking is not disabled on the machine specified by **-server**. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **bos stop** command sets the status flag for each server process specified by **-process** to **NotRun** in the **BosConfig** file on the server machine specified by **-server**; it then stops each process.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.bos** file on the machine specified by the **-server** option.

Implementation Specifics

On OS/390 DFS, the **bos stop** command stops processes controlled by the **bosserver**. These include the: **bakserver**, **flserver**, **ftserver**, **repserver**, **upserver**, and **upclient** processes. The remaining OS/390 DFS processes are controlled by the OS/390 system command **MODIFY**. For further information, see “modify dfs daemon” on page 329.

Related Information

Commands:

bos create	**bos shutdown**	**bos status**

File:

BosConfig

bos uninstall

Purpose

Installs the former versions of binary files.

Usage

The **bos uninstall** command replaces each binary file specified with the **-file** option with its next-oldest version. Use the **-server** option to specify the name of the server machine that houses the files to be manipulated. All specified files must reside in the same directory. By default, the command looks for the files in the **/opt/dfsglobal/bin** directory; use the **-dir** option to name a different directory. Versions of the files in other directories on the specified machine are not affected.

Important Note to Users

The **bos uninstall** command is not supported on OS/390 DFS.

bosserver

Purpose

Initializes the Basic OverSeer (BOS) Server process.

Format

bosserver [-adminlist *filename***] [-noauth] [-help]**

Options

-adminlist *filename*
: Specifies the file that contains principals and groups authorized to execute **bosserver** RPCs (usually using **bos** commands). If this option is omitted, the **bosserver** obtains the list of authorized users from the default administrative list file, **/opt/dcelocal/var/dfs/admin.bos**.

-noauth
: Invokes the **bosserver** with DFS authorization checking turned off. In this mode, DFS processes, including the **bosserver** process, do not check to see whether issuers have the necessary privilege to enter administrative commands.

  This option is intended for use when the BOS Server is initially installed on a server machine. Because it starts the **bosserver** with DFS authorization checking turned off, it allows the issuer to add members to the **admin.bos** administrative list and to add a key to the keytab file on the server machine.

  Use this mode sparingly, as it presents a security risk. Using this option forces all DFS server processes on the machine to run without DFS authorization checking.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

  The **help** and **apropos** commands available with all command suites are also available with **bosserver**. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The Basic OverSeer Server (BOS Server) monitors other DFS server processes, such as the **flserver** and **ftserver** processes, running on the machine and restarts failed processes automatically (without the intervention of a system administrator). The BOS Server, or **bosserver** process, monitors each server process that has a process entry in the local **BosConfig** file. The **bosserver** process must be run on all DFS server machines.

The first time the **bosserver** process is initialized, it creates several directories, such as the **/opt/dcelocal/var/dfs/adm** directory (and any nonexistent directories along this path), sets the owner to **root**, and sets the mode bits. On OS/390 DFS, **root** refers to a user with a **UID = 0**. The **bosserver** process also creates the **/opt/dcelocal/var/dfs/admin.bos** administrative list file and the **/opt/dcelocal/var/dfs/BosConfig** configuration file if either file does not already exist.

When it is started, the **bosserver** process also creates the **/opt/dcelocal/var/dfs/adm/BosLog** event log file if the file does not already exist. It then appends messages to the file. If the **BosLog** file exists when the **bosserver** process is started, the process moves it to the **BosLog.old** file in the same directory (overwriting the current **BosLog.old** file if it exists) before creating a new version to which to append messages.

The principals and groups in the **admin.bos** administrative list are authorized to issue BOS commands to stop, start, create, and modify server processes on that machine. For simplified administration, the same **admin.bos** administrative list can be used by all **bosserver** processes in the administrative domain.

When initially installing the BOS Server on a server machine, use the **-noauth** option to initialize the **bosserver** process with DFS authorization checking disabled. This creates the **NoAuth** file in the **/opt/dcelocal/var/dfs** directory on the local disk; when the file is present, DFS authorization checking is disabled on the machine.

With DFS authorization checking disabled, add members to the **admin.bos** list and add a key to the keytab file on the server machine. When these steps are complete, use the **bos setauth** command to enable DFS authorization checking. Because running with DFS authorization checking disabled is a serious security risk, enable DFS authorization checking as soon as the previous steps are complete. The **bos status** command can be used to determine whether DFS authorization checking is enabled or disabled on a machine; it displays the following message if DFS authorization checking is disabled on a machine (it does not display the message if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Output

If problems are encountered during initialization, the **bosserver** process displays error messages to the standard error output. The **bosserver** process keeps an event log in **/opt/dcelocal/var/dfs/adm/BosLog**.

Implementation Specifics

This process is part of the OS/390 DFS Base.

The **bosserver** process on OS/390 DFS is a load module in a partitioned data set (PDS), rather than a command.

The **bosserver** process on OS/390 DFS, by default, is started automatically by the DFS control task program, **DFSCNTL**. The **bosserver** process on OS/390 DFS runs under the control of the control task program, **DFSCNTL**. **DFSCNTL** and its **child** processes are controlled on OS/390 DFS by the OS/390 system command **MODIFY**. For further information, see “modify dfs daemon” on page 329.

Related Information

Commands:

bos setauth	**bos status**	

Files:

admin.bos	**BosLog**	**NoAuth**
BosConfig		

butc

Purpose

Initializes a Tape Coordinator process.

Format

butc [-tcid *tc_number*] **[-debuglevel** *trace_level*] **[-cell** *cellname*] **[-help]**

Options

-tcid *tc_number*
: Specifies the Tape Coordinator ID (TCID) associated with the Tape Coordinator to be initialized. The issuer of **bak** commands uses this number to indicate which Tape Coordinator is to execute a command.

  Legal values are the integers 0 to 1023. The value must match the value assigned to this Tape Coordinator's associated tape drive in the **TapeConfig** file. If this option is omitted, the default is 0 (zero).

-debuglevel *trace_level*
: Specifies the kinds of messages the Tape Coordinator produces in its monitoring window. The following two values are legal:

  - 0 (zero) indicates that the Tape Coordinator prompts the issuer only to place new tapes in the drive; the process does not report on its activities (other than to display some output as necessary for operations it executes). This is the default value.
  - 1 indicates that the Tape Coordinator reports on its activities as it restores filesets, in addition to prompting for new tapes as necessary.

-cell *cellname*
: Specifies the cell with respect to which the Tape Coordinator is to run. The Tape Coordinator communicates with the Backup Server in the specified cell. The Tape Coordinator can manipulate data in only the specified cell. A host entry must already be defined for the Tape Coordinator machine in the Backup Database of the specified cell.

  If this option is omitted, the default is the local cell of the issuer of the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

  The **help** and **apropos** commands available with all command suites are also available with **butc**. See “bos help” on page 504 and “bos apropos” on page 484 for more information on these commands.

Usage

The **butc** process starts a Tape Coordinator on a Tape Coordinator machine. Depending on the operations it executes, the **butc** process that runs as a result of this command contacts the Backup Database (by way of the Backup Server), the Fileset Location Database (by way of the Fileset Location Server), or Fileset Server processes.

The **butc** process also writes output to the following EBCDIC files in the directory **/opt/dcelocal/var/dfs/backup** on the local disk of the Tape Coordinator machine:

TL_*tapennn*
: The **TL_***tapennn* file (where *tapennn* is the device name of the tape drive with which the process is associated) is a log file that contains execution information about operations performed by the **butc** process. The level of detail to which each operation is described depends on the operation.

TE_*tapennn*
: The **TE_***tapennn* file (where *tapennn* is again the device name of the tape drive with which the process is associated) is an error file that contains information about problems encountered by the **butc** process.

The files contain similar information. For example, if you use the **bak dump** command to back up 100 filesets, the log file lists both the names of filesets that were successfully dumped to tape and the names of filesets that, for some reason, were omitted from the dump; the error file lists the names of only those filesets that were omitted from the dump.

Each time the **butc** process is started for a tape drive and Tape Coordinator pair, it automatically creates the two files. It then appends messages to the files as necessary. If the files already exist when the **butc** process is started, the process moves the current versions to files that end with **.old** extensions (for example, **TL_***tapennn***.old**) before creating new versions of the files to which to append messages. The process overwrites current **.old** files if they exist.

No maintenance is required for the log and error files associated with any tape drive; the files are created automatically or reinitialized each time the **butc** process for a tape drive and Tape Coordinator pair is started. However, the files should be browsed periodically to ensure that operations such as dumps and restores are completing without problems. For example, if a file cannot be dumped because a necessary Fileset Server or Fileset Location Server is unavailable at the time of the dump, the **butc** program writes an appropriate message to the log and error files.

Privilege Required

The issuer must have write and execute permissions on the **/opt/dcelocal/var/dfs/backup** directory, the directory in which the **butc** process creates its log and error files.

Implementation Specifics

This process is part of the OS/390 DFS Base.

The **butc** process on OS/390 DFS is a load module in a partitioned data set (PDS) rather than a command.

The **butc** process in OS/390 DFS, by default, is started manually. To start the **butc** process automatically, change the **ioepdcf** configuration type to **Y** for the **butc***nn* machine entry (for further information, see “ioepdcf” on page 382.).

The **butc** process on OS/390 DFS runs under the control of the control task program, **DFSCNTL**. **DFSCNTL** and its **child** processes are controlled on OS/390 DFS by the OS/390 system command, **MODIFY**. For further information, see “modify dfs daemon” on page 329.

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **TapeConfig** file is not necessary on OS/390 DFS and not available. This file may be available on another non-OS/390 Distributed File Service system.

Related Information

Command:

bak

Files:

TapeConfig	**TE_***tapennn*	**TL_***tapennn*

cm

Purpose

Introduction to the **cm** command suite.

Options

The following options are used with many **cm** commands. They are also listed with the commands that use them.

-path {*filename* | *directory_name*}
: Names the files, directories, or both to be used with the command.

-help
: Prints the online help for the command. All other valid options specified with this option are ignored. For complete details about receiving help, see "Receiving Help" on page 406.

Usage

The **cm** commands are issued by administrative users to update cached information on local workstations. Certain **cm** commands are available to all users to determine machine and cell information.

The files described in the following sections are used by the Cache Manager to determine its initial configuration and to store and track cached data. Each DFS client machine stores machine-specific versions of these files on its local disk.

The CacheInfo File

The **/opt/dcelocal/etc/CacheInfo** file specifies the Cache Manager's initial configuration. It is manually created during DFS client installation. The Cache Manager checks the file at initialization to determine certain cache configuration information.

Important Note to Users

On OS/390 DFS, the cache directory is a DCE Local File System aggregate. The minor device number of the DCE Local File System aggregate is specified in the second field of the **CacheInfo** file.

The **/opt/dcelocal/etc/CacheInfo** file on OS/390 DFS is stored as an EBCDIC file.

The file is a one-line EBCDIC file that contains three fields separated by colons. The fields provide the following information:

- The local directory where the Cache Manager mounts the DCE global namespace. The default is the global namespace designation (**/...**).
- The minor device number of the DCE Local File System aggregate to serve as the disk cache. The Cache Manager stores the **CacheItems**, **FilesetItems**, and **V***n* files in this cache.
- The size of the cache in 1024-byte (1kilobyte) blocks.

The CacheItems File

The **CacheItems** file is a binary file created and maintained by the Cache Manager and stored in the disk cache. The file records information such as the file ID number and data version number of each V file on a client machine using a disk cache.

The FilesetItems File

The **FilesetItems** file is a binary file created and maintained by the Cache Manager and stored in the disk cache. The file records the fileset-to-mount point mapping for each fileset accessed by the Cache Manager. The mappings allow the Cache Manager to respond correctly to commands such as **pwd**.

V Files

The **V***n* files, or V files, hold chunks of cached data on a client machine using a disk cache. In the name of an actual V file, *n* is an integer; each V file has a unique name (for example, **V1**, **V2**, and so on). The format of a V file depends on the information it contains.

By default, each V file holds up to 65,536 bytes (64 kilobytes) of data. The default size can be overridden with the an OS/390 UNIX System parameter and through the use of OS/390 DFS Environment Variables. For further information and recommendations, see Chapter 13, “Configuring the Cache Manager” on page 251.

Important Note to Users

On OS/390 DFS, you cannot access the **CacheItems**, **FilesetItems**, and **V***n* files through OS/390 UNIX. The DCE Local File System aggregate where these files reside is not locally mounted. Always use the commands provided with DFS to alter these files. For more information, see Chapter 13, “Configuring the Cache Manager” on page 251.

Cautions

Specific cautionary information is included with individual commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options:

$ **cm help**
: Displays a list of commands in a command suite.

$ **cm help** *command*
: Displays the syntax for a single command.

$ **cm apropos -topic** *string*
: Displays a short description of any commands that match the specified *string*.

Refer to “Receiving Help” on page 406 for complete information about the DFS help facilities.

Privilege Required

Specific privileges required by each command are listed with individual commands.

Implementation Specifics

On OS/390 DFS, the cache directory is a DCE Local File System aggregate. The minor device number of the DCE Local File System aggregate is specified in the second field of the **CacheInfo** file.

The **/opt/dcelocal/etc/Cacheinfo** file on OS/390 DFS is stored as an EBCDIC file.

On OS/390 DFS, you cannot access the **CacheItems**, **FilesetItems**, and **V***n* files through OS/390 UNIX. The DCE Local File System aggregate where these files reside is not locally mounted. Always use the commands provided with DFS to alter these files. For more information, see Chapter 13, “Configuring the Cache Manager” on page 251.

Related Information

Commands:

cm apropos	**cm getprotectlevels**	**cm setdevok**
cm checkfilesets	**cm getsetuid**	**cm setpreferences**
cm flush	**cm help**	**cm setprotectlevels**
cm flushfileset	**cm lscellinfo**	**cm setsetuid**
cm getcachesize	**cm lsstores**	**cm statservers**
cm getdevok	**cm resetstores**	**cm sysname**
cm getpreferences	**cm setcachesize**	**cm whereis**

Files:

CacheInfo	**FilesetItems**	**Vn**
CacheItems		

cm apropos

Purpose

Shows each help entry containing a specified string.

Format

cm apropos -topic *string* **[-help]**

Options

-topic *string*
: Specifies the keyword string for which to search. If it is more than a single word, surround the string with "" (double quotes) or other delimiters. Type all strings for **cm** commands in lower letters.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm apropos** command displays the first line of the online help entry for any **cm** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, us the **cm help** command.

Privilege Required

No privileges are required.

Output

The first line of an online help entry of a command names the command and briefly describes its function. This command displays the first line for any **cm** command where the string specified by **-topic** is part of the command name or the first line.

Examples

The following command lists all **cm** commands that have the word **cache** in their names or short descriptions:

```
$ cm apropos cache

flush: flush file data and status information from cache
getcachesize: get cache usage info
setcachesize: set cache size
```

Related Information

Command:
cm help

cm checkfilesets

Purpose

Forces the Cache Manager to update fileset-related information.

Format

cm checkfilesets [-help]

Options

-help	Prints the online help for this command.

Usage

The **cm checkfilesets** command forces the Cache Manager to discard its table of mappings between fileset names and fileset ID numbers. Because the Cache Manager needs the information in the table to fetch files, this command forces the Cache Manager to fetch the most recent information available about a fileset from the Fileset Location Server before the Cache Manager can fetch any more files from that fileset. (Usually, the Cache Manager flushes the table and constructs a new one every hour.)

This command is most useful if you know that a fileset name has changed or that there is a release of new read-only replicas. Issuing this command forces the Cache Manager to reference the fileset with the new name or the new read-only replica.

To force the Cache Manager to discard a cached file or directory, use the **cm flush** command. To force the Cache Manager to discard any data cached from filesets containing specified files or directories, use the **cm flushfileset** command.

Privilege Required

No privileges are required.

Related Information

Commands:

cm flush	**cm flushfileset**

cm flush

Purpose

Forces the Cache Manager to discard data cached from specified files or directories.

Format

cm flush [-path {*filename* **|** *directory_name***}...] [-help]**

Options

-path {*filename* **|** *directory_name***}**
Specifies each file or directory to be flushed. A file for which a full pathname is not specified is assumed to reside in the current working directory. In the case of a directory, all the name mappings and blocks associated with the directory are flushed; data cached from files or subdirectories that reside in the directory is not flushed. If this option is omitted, the current working directory is flushed.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm flush** command forces the Cache Manager to flush data cached from each file or directory specified with the **-path** option. All data cached from these files and directories is discarded. The next time the data is requested, the Cache Manager contacts the File Exporter to obtain the current version, along with new tokens and other associated status information.

This command does not discard any altered data in the cache not written to the central copy maintained by the File Exporter. It also does not affect data in the buffers of application programs.

It is also possible to flush all cached data that resides in the same fileset as a specific file or directory with the **cm flushfileset** command. To force the Cache Manager to update fileset-related information, use the **cm checkfilesets** command.

Privilege Required

No privileges are required.

Examples

The following command flushes the file **projectnotes**, which is in the current working directory, and all data from the subdirectory **plans** from the cache:

```
$ cm flush projectnotes plans/*
```

Related Information

Commands:

cm checkfilesets	**cm flushfileset**

cm flushfileset

Purpose

Forces the Cache Manager to discard data cached from filesets that contain specified files or directories.

Format

cm flushfileset [-path {*filename* **|** *directory_name***}...] [-help]**

Options

-path {*filename* **|** *directory_name***}**
: Specifies a file or directory from each fileset to be flushed. A file for which a full pathname is not specified is assumed to reside in the current working directory. If this option is omitted, the fileset containing the current working directory is flushed.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm flushfileset** command forces the Cache Manager to flush data cached from filesets that contain each file or directory specified with the **-path** option. All data cached from these filesets is discarded. The next time the data is requested, the Cache Manager contacts the File Exporter to obtain the current version, along with new tokens and other associated status information.

This command does not discard any altered data in the cache not written to the central copy maintained by the File Exporter. It also does not affect data in the buffers of application programs.

It is also possible to flush data cached from specific files or directories with the cm flush command. To force the Cache Manager to update fileset-related information, use the **cm checkfilesets** command.

Privilege Required

No privileges are required.

Examples

The following command flushes data cached from the fileset containing the current working directory and the directory **reports**, which are both at the same level in the file tree:

```
$ cm flushf . ../reports
```

Related Information

Commands:

cm checkfilesets   **cm flush**

cm getcachesize

Purpose

Shows current size of cache, the amount of cache in use, and the type of cache.

Format

cm getcachesize [-help]

Options

-help Prints the online help for this command.

Usage

The **cm getcachesize** command displays the current size of the cache available to the Cache Manager and the amount in use when the command is issued. It also displays the type of cache in use on the machine. The command works both on machines using disk caching and on machines using memory caching.

The information displayed by the command comes from the DFS Cache Manager colony address space of the machine where the command is issued. On machines using disk caching, the current cache size may disagree with the default setting specified in the **CacheInfo** file if the cache size was set with the **cm setcachesize** command. Regardless of the type of caching (disk or memory) in use, the size may also disagree with the default setting if it was changed through the use of OS/390 DFS environment variables (see Chapter 13, “Configuring the Cache Manager” on page 251).

Privilege Required

No privileges are required.

Output

The **cm getcachesize** command displays the following output:

```
DFS is using amount of the available size 1K byte type cache blocks.
```

In the output,

- *amount* is the number of kilobyte blocks the Cache Manager is currently using.
- *size* is the total number of kilobyte blocks available to the Cache Manager (the current cache size).
- *type* is the type of cache (disk or memory) in use on the machine.

Examples

The following command shows the output on a machine with a 25,000 kilobyte disk cache:

```
$ cm getcachesize

DFS is using 22876 of the available 25000 1K byte disk cache blocks.
```

Related Information

Command:
cm setcachesize
File:
CacheInfo

cm getdevok

Purpose

Shows whether device files from specified filesets are honored by the Cache Manager.

Format

cm getdevok [-path {*filename* **|** *directory_name***}...] [-help]**

Options

-path {*filename* **|** *directory_name***}**
: Names a file or directory from each fileset whose device file status information is to be displayed. If this option is omitted, status information is displayed for the fileset containing the current working directory.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm getdevok** command reports whether the Cache Manager honors device files that reside in the indicated filesets. Indicate each fileset for which you want device file status information is desired is indicated by specifying the name of a file or directory on the fileset with the **-path** option. This information comes from the kernel of the workstation where the command is issued.

System administrators set whether device files are to be honored on a per-fileset and per-Cache Manager basis with the **cm setdevok** command. By default, the Cache Manager does not honor device files from a fileset. (The kernel always honors device files stored in the **/dev** directory.)

Privilege Required

No privileges are required.

Output

The **cm getdevok** command first displays the following line:

`Fileset` *pathname* `status:`

In the output, *pathname* is the name of a file or directory specified with the **-path** option. For each specified file or directory, the following output values are possible for the fileset on which it resides:

`device files allowed.`
: Indicates that device files from the fileset are honored.

`device files not allowed.`
: Indicates that device files from the fileset are not honored.

`cm: the fileset on which '`*`pathname`*`' resides does not exist.`
: Indicates the specified pathname is invalid.

Examples

The following command indicates that device files from the fileset that contains the directory **/.../abc.com/fs/usr/jlw** are not honored by the Cache Manager:

```
$ cm getdevok /.../abc.com/fs/usr/jlw

/.../abc.com/fs/user/jlw status: device files not allowed.
```

Related Information

Command:
cm setdevok

cm getpreferences

Purpose

Displays the Cache Manager preferences for File Server machines.

Format

cm getpreferences [-path *filename***] [-numeric] [-fldb] [-help]**

Options

-path *filename*
: Specifies the full pathname of a file to which the command is to write the server preferences that it reports. If the specified file already exists, the command overwrites it. The command fails if the specified pathname names a directory. Omit this option to display the preferences on standard output (**stdout**).

-numeric Directs the command to display the IP addresses rather than the hostnames of the File Server machines in the server preferences that it reports. Omit this option to display the hostname (for example, **fs1.abc.com**) of each machine.

-fldb Displays the Cache Manager preferences for Fileset Location Server (flserver) machines instead of for File Server machines.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm getpreferences** command displays the Cache Manager preferences for File Server machines. A preference consists of the hostname or IP address of a File Server machine followed by the machine rank. The rank is an integer in the range from 1 to 65,534. Each Cache Manager maintains its own local collection of preferences, so two Cache Managers can have two different preferences for the same File Server machines.

A File Server machine rank determines the Cache Manager preference for accessing data from read-only filesets that reside on that server. When the Cache Manager must access data from the read-only version of a fileset, it determines which File Server machines house a read-only replica of the fileset. If it determines that multiple File Server machines house a replica of the fileset, the Cache Manager attempts to access the replica on the machine that has the lowest recorded rank.

If it cannot access the replica on the File Server machine that has the lowest rank (possibly because the machine or the network on which the machine resides is down), the Cache Manager attempts to access the replica on the machine that has the next-lowest rank. It continues in this manner until it either accesses the replica or determines that all of the File Server machines on which the replica resides are unavailable.

The Cache Manager records a rank for each File Server machine that it contacts or that houses a replica of a read-only fileset from which it has accessed data. It also records a rank for each File Server machine for which a rank has been specified with the **cm setpreferences** command. The Cache Manager stores the IP addresses and ranks in the kernel of the local machine. It makes no distinction between preferences for machines from the local cell and those for machines from a foreign cell.

The **cm getpreferences** command displays information on standard output by default. Use the **-path** option to specify the complete pathname of a file to which the command is to write its output. If you include **-path** option, the command displays no output on standard output.

Cache Managers also maintain preferences for Fileset Location Server (flserver) machines. To display the preferences for Fileset Location Server machines instead of for File Server machines, specify the **-fldb** option.

Privilege Required

No privileges are required.

Output

The **cm getpreferences** command displays a separate line of output for each File Server machine for which it maintains a preference. By default, each line consists of the name of a File Server machine followed by the machine rank, as follows:

```
hostname          rank
```

where *hostname* is the name of the File Server machine, and *rank* is the rank associated with the machine. If the **-numeric** option is included with the command, the command displays the IP address, in dotted decimal format, of each File Server machine instead of the machine name. The command also displays the IP address of any machine whose name it cannot determine (for example, if a network outage prevents it from resolving the address into the name). If the **-fldb** option is specified, the output refers to Fileset Location Server machines instead of to File Server machines.

Examples

The following command displays the preferences (the list of File Server machines and their respective ranks) associated with the Cache Manager on the local machine. The local machine belongs to the DCE cell named **dce.abc.com**; the ranks of the File Server machines from the **dce.abc.com** cell are lower than the ranks of the File Server machines from the foreign cell, **dce.def.com**. The command shows the IP addresses, not the names, of two machines from the foreign cell because it cannot currently determine their names.

```
$ cm getp

fs2.abc.com             20007
fs3.abc.com             30002
fs1.abc.com             20011
fs4.abc.com             30010
server1.def.com         40002
121.86.33.34            40000
server6.def.com         40012
121.86.33.37            40005
```

The following command displays the same Cache Manager preferences, but the **-numeric** option is included with the command to display the IP addresses rather than the names of all File Server machines. The IP address of the local machine is **128.21.16.221**. The two File Server machines on the same subnetwork as the local machine have ranks of 20,007 and 20,011; the two File Server machines on a different subnetwork in the same network as the local machine have ranks of 30,002 and 30,010; the remainder of the File Server machines are in a different network, so their ranks range from 40,000 to 40,012.

```
$ cm getp -n

128.21.16.214        20007
128.21.18.99         30002
128.21.16.212        20011
128.21.18.100        30010
121.86.33.41         40002
121.86.33.34         40000
121.86.33.36         40012
121.86.33.37         40005
```

Related Information

Command:
cm setpreferences

cm getprotectlevels

Purpose

Returns the current DCE RPC authentication level settings for communications between the Cache Manager and File Servers.

Format

cm getprotectlevels [-help]

Options

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm getprotectlevels** command returns the current Cache Manager DCE RPC authentication level settings. The returned values include separate local and foreign cell settings for the initial and minimum authentication levels for communications with File Servers.

The Cache Manager and File Server default settings are such that communications occur at the Packet authentication level for the local cell.

The authentication bounds for the File Server itself are set through the **fxd** command. In addition to a general pair of upper and lower bounds for all communications between the File Server and Cache Manager, administrators can also set advisory bounds on a per fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level. Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset. You can display the current advisory RPC authentication bounds for a fileset through either the **fts lsfldb** or **fts lsft** commands.

Privilege Required

No privileges are required.

Output

The output consists of the following four lines:

```
Initial protection level in the local cell: level
Minimum protection level in the local cell: level
Initial protection level in non-local cells: level
Minimum protection level in non-local cells: level
```

where *level* is one of the various DCE RPC authentication levels, whose possible values are:

- **rpc_c_protect_level_default** - default : Use the DCE default authentication level.
- **rpc_c_protect_level_none** - none : Perform no authentication.
- **rpc_c_protect_level_connect** - connect : Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_c_protect_level_call** - call : Authenticate only at the beginning of each RPC received.

- **rpc_c_protect_level_pkt** - packet : Ensure that all data received is from the expected principal.
- **rpc_c_protect_level_pkt_integ** - packet integrity : Authenticate and verify that none of the of the data transferred has been modified.
- **rpc_c_protect_level_pkt_privacy** - packet privacy : Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.
- **rpc_c_protect_level_cdmf_priv** - cdmf privacy : Perform authentication as specified by all the previous levels and also encrypt each RPC argument value using a lower level of encryption than packet privacy.

Examples

The following command returns the current authentication levels for communications between the Cache Manager and Files Servers:

```
$ cm getprotectlevels

Initial protection level in the local cell: rpc_c_protect_level_pkt
Minimum protection level in the local cell: rpc_c_protect_level_none
Initial protection level in non-local cells: rpc_c_protect_level_pkt_integ
Minimum protection level in non-local cells: rpc_c_protect_level_pkt
anonymous access to untrusted cells is permitted
```

Related Information

Command:

cm setprotectlevels	**fxd**	**fts setprotectlevels**
dfsd		

cm getsetuid

Purpose

Shows the status of **setuid** programs from specified filesets.

Format

cm getsetuid [path {*filename* **|** *directory_name***}...] [-help]**

Options

-path {*filename* **|** *directory_name***}**
: Names a file or directory from each fileset whose **setuid** permission is to be displayed. If this option is omitted, permission information is displayed for the fileset containing the current working directory.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm getsetuid** command reports whether the Cache Manager allows **setuid** programs from the indicated filesets to run with **setuid** permission. Each fileset whose **setuid** permission is desired is indicated by specifying the name of a file or directory on the fileset with the **-path** option. This information comes from the kernel of the workstation where the command is issued.

Note that **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

Because **setgid** programs on filesets are enabled or disabled along with **setuid** programs, this command also reports the status of **setgid** programs on the indicated filesets. System administrators set **setuid** and **setgid** status on a per-fileset and per-Cache Manager basis with the **cm setsetuid** command. By default, the Cache Manager does not allow **setuid** programs from a fileset to execute with **setuid** permission.

Privilege Required

No privileges are required.

Output

The **cm getsetuid** command first displays the following line:

`Fileset` *pathname* `status:`

In the output, *pathname* is the name of a file or directory specified with the **-path** option. For each specified file or directory, the following output values are possible for the fileset on which it resides:

`setuid allowed.`
: Indicates that **setuid** and **setgid** programs from the fileset are enabled.

`no setuid allowed.`
: Indicates that **setuid** and **setgid** programs from the fileset are disabled.

`cm: the fileset on which '`*`pathname`*`' resides does not exist.`
: Indicates the specified pathname is invalid.

Examples

The following command indicates that **setuid** and **setgid** programs from the fileset that contains the directory **/.../abc.com/fs/usr/jlw** are disabled:

```
$ cm getsetuid /.../abc.com/fs/usr/jlw

Fileset /.../abc.com/fs/usr/jlw status: no setuid allowed.
```

Related Information

Command:
cm setsetuid

cm help

Purpose

Shows the syntax of specified **cm** commands or lists functional descriptions of all **cm** commands.

Format

cm help [-topic *string***] [-help]**

Options

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **flush**, not **cm flush**). If this option is omitted, the output provides a short description of all **cm** commands.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm help** command displays the first (name and short description) of the online help entry for every **cm** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **cm apropos** command to show each help entry a specified string.

Privilege Required

No privileges are required.

Output

The online help entry for each **cm** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Examples

The following command displays the online help entry for the **cm flush** command:

```
$ cm help flush

cm flush: flush file from cache
Usage: cm flush [-path {filename | directory_name}...] [-help]
```

Related Information

Command:
cm apropos

cm lscellinfo

Purpose

Shows database server machines in cells known to the Cache Manager.

Format

cm lscellinfo [-help]

Options

-help Prints the online help for this command.

Usage

The **cm lscellinfo** command formats and displays the Cache Manager's list of Fileset Location Database (FLDB) machines in its home cell and any foreign cells the Cache Manager has accessed. The information displayed by the command comes from the DFS Cache Manager colony address space of the machine where the command is issued.

Privilege Required

No privileges are required.

Output

The output contains one line for the local cell and one line for each cell that the Cache Manager has accessed. Each cell name is followed by a list of its database server machines (referred to as **hosts**). In a multihomed server environment (an FLDB machine can have up to four IP addresses listed in the Cache Manager's preferencees), *hosts* corresponds to the IP addresses or host names that the Cache Manager is currently using to access each particular FLDB machine. Therefore, the command output lists only one machine name for each FLDB machine.

Examples

The following command shows output for several cells:

```
$ cm lscellinfo

Cell abc.com on host fs2.abc.com.
Cell state.edu on host fs11.fs.state.edu.
```

cm lsstores

Purpose

Lists filesets that contain data the Cache Manager cannot write back to a File Server machine.

Format

cm lsstores [-help]

Options

-help Prints the online help for this command.

Usage

The **cm lsstores** command lists the fileset ID numbers of filesets that contain data the Cache Manager cannot write back to a File Server machine. The information displayed by the command comes from the DFS Cache Manager colony address space of the machine where the command is issued.

On occasion, a file server machine may be unavailable to the Cache Manager (possibly because the files server machine is down or because a network problem prevents the Cache Manager from contacting the machine). In such cases, the Cache Manager cannot write data back to the file server machine. The Cache Manager displays a message on the screen to notify the user that it cannot write the data to the unavailable machine. If possible, it also returns a failure code to the application program using the data.

The Cache Manager keeps the unstored data in the cache and continues to attempt to contact the File Server machine until it can store the data. (the frequency with which the Cache Manager attempts to reach a File Server machine is defined with the **-pollinterval** option of **fxd** 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.

Important Note to Users

On OS/390 DFS, **fxd** is run as part of the **dfskern** process in the DFS server address space. All options for **fxd** can be specified in the **ioepdcf** file. For further information, see "ioepdcf" on page 382.

Once the problem is alleviated, the Cache Manager can reach the File Server machine and store the data.

The Cache Manager discards unstored data only when:

- It needs to make room in the cache for other data. Given an average-sized cache with average usage, the Cache Manager rarely needs to discard unstored data.
- The **cm resetstores** command is issued to force it to discard unstored data from the cache.

Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

Privilege Required

No privileges are required.

Output

If the Cache Manager cannot store data to one or more filesets, the command displays the fileset ID number of each fileset to which data cannot be stored. If the Cache Manager has been able to store all data, the command displays the following message:

```
There are no filesets currently being waited on.
```

Implementation Specifics

On OS/390 DFS, **fxd** is run as part of the **dfskern** process in the DFS server address space. All options for **fxd** can be specified in the **ioepdcf** file. For further information, see “ioepdcf” on page 382. This command is part of OS/390 DFS.

Related Information

Command:
cm resetstores

cm resetstores

Purpose

Cancels attempts by the Cache Manager to contact unavailable File Server machines and discards all data the Cache Manager cannot store to such machines.

Format

cm resetstores [-help]

Options

-help Prints the online help for this command.

Usage

The **cm resetstores** command cancels the Cache Manager's continued attempts to contact unavailable File Server machines. *All* data the Cache Manager cannot store is discarded; there is no way to selectively discard individual files or data from specific filesets.

Occasionally, a File server machine may be unavailable to the Cache Manager (possibly because the File Server machine is down or because a network problem prevents the Cache Manager from contacting the machine). In such cases, the Cache Manager cannot write data back to the File Server machine. The Cache Manager displays a message on the screen to notify the user that it cannot write the data to the unavailable machine. If possible, it also returns a failure code to the application program using the data.

The Cache Manager keeps the unstored data in the cache and continues to attempt to contact the File Server machine until it can store the data. (The frequency with which it attempts to reach a File Server machine is defined with the **-pollinterval** option of **fxd** issued on that File Server machine.) In the meantime, the user can take corrective measures to alleviate the problem that prevents the data from being stored; for example, the File Server machine can be restarted.

Important Note to Users

On OS/390 DFS, **fxd** is run as part of the **dfskern** process in the DFS server address space. All options for **fxd** can be specified in the **ioepdcf** file. For further information, see “ioepdcf” on page 382.

Once the problem is alleviated, the Cache Manager can reach the File Server machine and store the data.

The Cache Manager discards unstored data only when:

- It needs to make room in the cache for other data. Given an average-sized cache with average usage, the Cache Manager rarely needs to discard unstored data.
- The **cm resetstores** command is issued to force it to discard unstored data from the cache.

Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

Note that the **cm resetstores** command affects only data that could not be stored to a File Server machine; it does not affect the other data in the cache. Nonetheless, be cautious when issuing the **cm resetstores** command: Issue the **cm lsstores** command first to list the fileset ID numbers of filesets that contain data the Cache Manager cannot write to a File Server machine; examine the output of the command to be sure that you know from which filesets unstored data will be discarded (you may also be able to use this information to ensure that unstored data from the indicated filesets can be safely discarded).

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Related Information

Command:
cm lsstores

cm setcachesize

Purpose

Sets the size of a disk cache.

Format

cm setcachesize {-size *kbytes* **| -reset} [-help]**

Options

-size *kbytes*
: Specifies the number of 1-kilobyte blocks the Cache Manager can use for the cache. The smallest allowable value is 1. Specifying a value of 0 (zero) sets the cache size to the default specified in the **CacheInfo** file. Use this option or use **-reset**.

-reset
: Returns the cache size to the value set when the Cache Manager was last started. Use this option or use **-size**.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm setcachesize** command changes the amount of local disk space the Cache Manager uses for its data cache. Specify a number of kilobyte blocks. Do not set the cache size to exceed 75% of the actual disk space available for the cache. *Do not use this command on a machine using a memory cache.*

The cache size cannot be set to a value less than twice the value of the chunk size in use by the Cache Manager. If a value smaller than twice the chunk size is specified with the **-size** option, the following message is displayed:

path: `Cache size of` *size* `is too small; value was rounded up.`

In the message, *path* is the specified path to the **cm** program (usually just **cm**) and *size* is the size, in kilobytes, specified with the command. The standard message reporting the new cache size (the size to which the cache was rounded) is then displayed; see “Output” on page 568 for an example of the message.

To return the cache size to the default value specified in the **CacheInfo** file, specify 0 (zero) as the number of kilobyte blocks with the **-size** option. To return the cache size to the value set when the Cache Manager was last started, use the **-reset** option instead of the **-size** option; the **-reset** option also sets the size to the amount specified in the **CacheInfo** file unless the **-blocks** option was used with the **dfsd** command to override the **CacheInfo** value, in which case the value set with the **dfsd** command is used.

The **cm getcachesize** command displays the current cache size and the amount of space in use for both disk and memory caches. It also reports the type of cache (disk or memory) in use.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Output

The following message is displayed whenever this command is used to set the cache size:

path: `New cache size set:` *size*.

In the message, *path* is the specified path to the **cm** program (usually just **cm**) and *size* is the new cache size, in kilobytes.

Examples

The following command sets the cache size to 25,000 kilobyte blocks:

```
$ cm setca 25000

cm: New cache size set: 25000.
```

The following command resets the cache size to the value set when the Cache Manager was last started (50,000 kilobyte blocks, in this case):

```
$ cm setca -r

cm: New cache size set: 50000.
```

Related Information

Command:
cm getcachesize
File:
CacheInfo

cm setdevok

Purpose

Specifies whether device files on specified filesets are honored by the Cache Manager.

Format

cm setdevok [-path {*filename* **|** *directory_name***}...] [-state {on | off}] [-help]**

Options

-path {*filename* **|** *directory_name***}**
: Names a file or directory from each fileset for which device file status is to be changed. If this option is omitted, the status is changed for the fileset containing the current working directory.

-state
: Specifies whether device files from the filesets indicated with **-path** are to be honored. Specify **on** with this option to honor device files from the indicated filesets; specify **off** with this option to prevent device files from the indicated filesets from being honored. If this option is omitted, device files from the filesets are honored (the command has no effect if device files were already honored).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

This command specifies whether device files from the indicated filesets are honored by the Cache Manager. Indicate each fileset whose device files are to be honored or not honored by specifying the name of a file or directory on the fileset with the **-path** option. Device files are honored on a per-fileset and per-Cache Manager basis. This command is commonly included in a start-up file (**/etc/rc** or its equivalent) to honor device files at machine startup.

If **on** is specified with the **-state** option, or if the **-state** option is omitted, the Cache Manager honors device files from the indicated filesets. If **off** is specified with the **-state** option, the Cache Manager does not honor device files from the indicated filesets. By default, the Cache Manager does not honor device files from a fileset. (The kernel always honors device files stored in the **/dev** directory.)

The **cm getdevok** command displays whether the Cache Manager honors device files from indicated filesets.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Examples

The following command causes device files that reside on the fileset that contains the directory **/.../abc.com/fs/usr/jlw** to be honored:

```
$ cm setdevok /.../abc.com/fs/usr/jlw
```

Related Information

Command:
cm getdevok

cm setpreferences

Purpose

Set the Cache Manager's preferences for File Server machines.

Format

cm setpreferences [-server *machine rank ...***] [-path** *filename***] [-stdin] [-fldb] [-help]**

Options

-server *machine rank ...*
: Specifies one or more pairs of File Server machines and their respective ranks. Separate each machine specification and each rank with one or more spaces. Refer to **Specifying Preferences** for information about specifying File Server machines and ranks.

-path *filename*
: Specifies the full path name of a file from which the command is to read pairs of File Server machines and their respective 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. Refer to **Specifying Preferences** for information about specifying File Server machines and ranks.

-stdin
: Directs the command to read pairs of File Server machines and their respective ranks from standard input (**stdin**). Separate each machine specification and each rank with one or more spaces. Refer to **Specifying Preferences** for information about specifying File Server machines and ranks.

-fldb
: Specifies that the command applies to Fileset Location Server (flserver) machines instead of to File Server machines.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm setpreferences** command sets the Cache Manager's preferences for one or more File Server machines. A preference consists of the hostname or IP address of a File Server machine followed by the machine's numerical rank. A File Server machine's rank determines the Cache Manager's preference for electing to access read-only replicas that reside on the File Server machine over replicas that reside on other machines.

When the Cache Manager needs to access data from the read-only version of a fileset, it first contacts the Fileset Location (FL) Server to determine the names of the File Server machines on which a read-only replica of the fileset resides. If multiple File Server machines house a replica of the fileset, the Cache Manager consults its preferences for the File Server machines and attempts to access the replica on the server that has the lowest recorded rank. If multiple File Server machines have the same rank, the Cache Manager selects them in the order in which it received their names from the Fileset Location Server.

If it cannot access the replica on the File Server machine with the lowest rank (possibly because the machine or the network on which the machine resides is down), the Cache Manager attempts to access the replica on the machine that has the next-lowest rank. It continues in this manner until it either succeeds in accessing the replica or determines that all of the File Server machines on which the replica is stored are unavailable.

The Cache Manager stores its preferences in the kernel of the local machine. The preferences are lost each time the Cache Manager is initialized with the **dfsd** command (each time the client machine is rebooted). After it is initialized, the Cache Manager rebuilds its collection of preferences by assigning a rank to each File Server machine that it contacts or that houses a replica of a read-only fileset from which it accesses data. The Cache Manager makes no distinction between preferences for File Server machines from the local cell and those for File Server machines from a foreign cell.

You can use the **cm setpreferences** command to define or change the rank associated with a File Server machine. If the command specifies a rank for a File Server machine for which the Cache Manger has no rank, the command defines the machine's initial rank; if the command specifies a rank for a machine for which the Cache Manager already has a rank, the command changes the current rank to match the rank you specify. You can include the **cm setpreferences** command in a machine's initialization file to load a predefined collection of server preferences when the machine is rebooted.

Default preferences bias the Cache Manager to select read-only filesets from the File Server machines that are in the same subnetwork or network as the local machine. You can use the **cm setpreferences** command to alter the default preferences. Because each Cache Manager maintains its own local collection of File Server preferences, two Cache Managers can have two different ranks for the same File Server machine.

Cache Managers also maintain preferences for Fileset Location Server (flserver) machines. Include the **-fldb** option on the **cm setpreferences** command to specify that the command applies to Fileset Location Server machines. When the **-fldb** option is used, machines that are specified with the **-server** option or that are listed in the file named by the **-path** option refer to Fileset Location Server machines, not File Server machines.

Specifying Preferences

Using the **cm setpreferences** command, you specify preferences as pairs of values. The first value of the pair is the hostname (for example, **fs1.abc.com**) or IP address, in dotted decimal format, of a File Server machine; the second value of the pair is the machine's numerical rank, an integer in the range from 1 to 65,534.

To minimize the chances that multiple File Server machines are assigned the same rank and to ensure maximum load balancing among File Server machines, the Cache Manager adds a random number in the range from 0 (zero) to 15 to each rank that you specify. For example, if you specify a rank of 15,000, the Cache Manager records the rank as an integer in the range from 15,000 to 15,015.

You can specify pairs of File Server machines and their ranks

- On the command line by using the **-server** option. Use the option to tune the preferences manually in response to system or network adjustments.
- From a file by using the **-path** option. Use the option to configure one or more Cache Managers with a fixed set of preferences, specifying a file created manually or generated automatically. You can use the **cm getpreferences** command to generate a file of preferences that has the proper format.
- From standard input by using the **-stdin** option. Use the option to pipe preferences to the command from a user-defined process that generates preferences in an acceptable format.

The **-server**, **-path**, and **-stdin** options are not mutually exclusive. You can include any combination of these options with the command to provide input from multiple sources. Note that the command does not verify hostnames or IP addresses specified with any of its options. You can add a preference for an invalid hostname or IP address; the Cache Manager stores such preferences, but it ignores them (the Cache Manager never needs to consult such preferences).

Allowing the Cache Manager to Assign Preferences

The Cache Manager bases default ranks that it calculates on IP addresses rather than on actual physical considerations such as location or distance. It uses the following heuristic to calculate default ranks for File Server machines:

- If the local machine is also a File Server machine, the machine receives an initial rank of 5000.
- Each File Server machine in the same subnetwork as the local machine receives an initial rank of 20,000.
- Each File Server machine in the same network as the local machine receives an initial rank of 30,000.
- Each File Server machine in a different network from the local machine receives an initial rank of 40,000.
- Each File Server machine for which the Cache Manager can determine no network information receives an initial rank of 40,000.

As it does with ranks specified with the **cm setpreferences** command, the Cache Manager adds a random number in the range from 0 (zero) to 15 to each initial rank that it determines. For example, while it assigns an initial rank of 20,000 to a File Server machine in the same subnetwork as the local machine, the Cache Manager records the actual rank as an integer in the range from 20,000 to 20,015.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Output

By default, the **cm setpreferences** command displays no output.

Examples

The following command uses the **-server** option to set the Cache Manager's preferences for the File Server machines named **fs3.abc.com** and **fs4.abc.com**, the latter of which is specified by IP address **128.21.18.100**. The two File Server machines reside in a different subnetwork in the same network as the local machine, so the Cache Manager would assign each a rank of 30,000 plus an integer in the range from 0 to 15. To make the Cache Manager prefer these File Server machines over File Server machines in other subnetworks in the local network, the **cm setpreferences** command is used to assign these machines ranks of 25,000 to which the Cache Manager adds an integer in the range from 0 to 15.

```
# cm setp -se fs3.abc.com 25000 128.21.18.100 25000
```

The following command uses the **-server** option to set the Cache Manager's preferences for the same two File Server machines, but it also uses the **-path** option to read a collection of preferences from a file that resides on the local machine at **/etc/cm.prefs**:

```
# cm setp -se fs3.abc.com 25000 128.21.18.100 25000 -p /etc/cm.prefs
```

The file **/etc/cm.prefs** has the following contents and format:

```
128.21.16.214 7500
128.21.16.212 7500
121.86.33.41 39000
121.86.33.34 39000
121.86.33.36 41000 121.86.33.37 41000
```

The following command uses the **-stdin** option to read preferences from standard input. The preferences are piped to the command from a program, **calc_prefs**, which was written by the issuer to calculate preferences based on values significant to the local cell.

```
# calc_prefs | cm setp -stdin
```

Related Information

Commands:

cm getpreferences	**dfsd**

cm setprotectlevels

Purpose

Adjusts DCE remote procedure call (RPC) authentication levels for communications between the Cache Manager and File Servers.

Format

cm setprotectlevels [-initiallocalprotectlevel *level***] [-minlocalprotectlevel** *level***] [-initialremoteprotectlevel** *level***] [-minremoteprotectlevel** *level***] [-anonaccess {on | off}] [-help]**

Options

-initiallocalprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage."

-minlocalprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage."

-initialremoteprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage."

-minremoteprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage."

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

-anonaccess
: Allows access to untrusted cells.

Usage

The **cm setprotectlevels** command adjusts the DCE RPC security level for RPCs sent between a Cache Manager and DFS File Servers. The command adjusts two levels: an initial DCE RPC security level used as a starting point in security level negotiations between the Cache Manager and a File Server and the minimum DCE RPC security level the Cache Manager will accept for such communications. Two sets of these levels are maintained: one set specifies the security levels for communications with File Servers within the local cell and the other set specifies the security levels for communications with File Servers within foreign cells. Both sets of security levels are initially set through the **dfsd** command options (specified in the DFSCM **_IOE_CM_PARMS** environment variable, refer to "Cache Manager Initialization Parameters" on page 273).

In operation, the Cache Manager and File Server interact to arrive at a mutually acceptable authentication level for communications. The negotiation starts with an RPC using the initial authentication level sent from the Cache Manager to the File Server. If the initial authentication level is outside the minimum or maximum bounds set at the File Server, the File Server returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Server requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Server). Once the Cache Manager and File Server have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Server.

The authentication bounds for communications at the File Server itself is set through the **fxd** command. The Cache Manager and **fxd** default settings are such that communications occur at the Packet authentication level for the local cell.

In addition to a general pair of upper and lower bounds for all communications between the File Server and Cache Manager, administrators can also set advisory bounds on a per fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level (they may be enforced in a future version of DFS). Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset.

Note that the use of this command does not preclude communications with File Servers running earlier versions of DFS.

The various authentication levels are set by specifying either an integer value between 0 and 7, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **0** or **default** or **rpc_c_protect_level_default**: Use the DCE default authentication level.
- **1** or **none** or **rpc_c_protect_level_none**: Perform no authentication.
- **2** or **connect** or **rpc_c_protect_level_connect**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **3** or **call** or **rpc_c_protect_level_call**: Authenticate only at the beginning of each RPC received.
- **4** or **pkt** or **rpc_c_protect_level_pkt**: Ensure that all data received is from the expected host.
- **5** or **pkt_integrity** or **rpc_c_protect_level_pkt_integrity**: Authenticate and verify that none of the data transferred has been modified.
- **6** or **pkt_privacy** or **rpc_c_protect_level_pkt_privacy**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.
- **7** or **cdmf_priv** or **rpc_c_protect_level_cdmf_priv**: Perform authentication as specified by all of the previous levels (except 6) and also encrypt each RPC argument value. This level provides a lower level of packet privacy than **rpc_c_protect_level_pkt_privacy**.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

Privilege Required

The issuer must be logged in as **root** on the local machine.

Examples

The following command sets the following authentication values:

1. The initial authentication level for communications with File Servers in the local cell is set to packet integrity.
2. The minimum authentication level for communications with File Servers in the local cell is set to packet.
3. The initial authentication level for communications with File Servers in foreign cells is set to packet privacy.
4. The minimum authentication level for communications with File Servers in foreign cells is set to packet privacy.

```
$ cm setprotectlevels -initiallocalprotectlevel 5 -minlocalprotectlevel 4 \
                      -initialremoteprotectlevel 6 -minremoteprotectlevel 6
```

Related Information

Command:

cm getprotectlevels	**fts setprotectlevels**	**fxd**
dfsd		

cm setsetuid

Purpose

Enables or disables **setuid** programs from specified filesets.

Format

cm setsetuid [-path {*filename* **|** *directory_name***}...] [-state {on | off}] [-help]**

Options

-path {*filename* **|** *directory_name***}**
: Names a file or directory from each fileset whose **setuid** status is to be changed. If this option is omitted, the status is changed for the fileset containing the current working directory.

-state
: Allows or disallows **setuid** programs from the filesets indicated with **-path** to execute with **setuid** permission. Specify **on** with this option to allow **setuid** programs from the indicated filesets to execute with **setuid** permission; specify **off** with this option to disallow **setuid** programs from the indicated filesets to execute with **setuid** permission. If this option is omitted, **setuid** programs from the filesets are allowed to execute with **setuid** permission (the command has no effect if **setuid** permission was already enabled).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm setsetuid** command enables **setuid** programs from the indicated filesets to execute with **setuid** permission or prevents them from executing with **setuid** permission. Indicate each fileset whose **setuid** permission is to be enabled or disabled is indicated by specifying the name of a file or directory on the fileset with the **-path** option. The permissions are enabled or disabled on a per-fileset and per-Cache Manager basis. This command is commonly included in a start-up file (**/etc/rc** or its equivalent) to enable **setuid** programs at machine startup.

If **on** is specified with the **-state** option, or if the **-state** option is omitted, the Cache Manager allows **setuid** programs from the indicated filesets to execute with **setuid** permission. If **off** is specified with the **-state** option, the Cache Manager does not allow **setuid** programs from the indicated filesets to execute with **setuid** permission. By default, the Cache Manager does not allow **setuid** programs to execute with **setuid** permission.

A **setuid** program is indicated by setting a mode bit associated with an executable file. While a **setuid** program executes, the person executing the program is treated as the owner of the program. The effective User Identification Number (UID) of the executing program is the UID of the person who owns the program, not the UID of the person who initiated the program's execution. Thus, the person executing the program is granted the same permissions as the person who owns the program for the duration of the program's execution.

The **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

The **cm setsetuid** enables or disables **setgid** programs from the indicated filesets at the same time that it changes the status of **setuid** programs. The **cm getsetuid** command displays whether the Cache Manager allows **setuid** and **setgid** programs from indicated filesets to execute.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Examples

The following command enables **setuid** and **setgid** programs that reside on the fileset containing the directory **/.../abc.com/fs/usr/jlw**:

```
$ cm setsetuid /.../abc.com/fs/usr/jlw
```

Related Information

Command:
cm getsetuid

cm statservers

Purpose

Checks the status of File Server machines.

Format

cm statservers [{-cell *cellname* **| -all}] [-fast] [-help]**

Options

-cell *cellname*
: Specifies the name of the specific cell the Cache Manager is to probe for the status of each File Server machine it has contacted or has attempted to contact from that cell. The Cache Manager probes only machines in the specified cell. Use this option or use the **-all** option; omit both options to direct the Cache Manager to probe only machines in the local cell.

-all
: Directs the Cache Manager to probe all of the machines it has contacted in all cells. Use this option or use the **-cell** option; omit both options to direct the Cache Manager to probe only machines in the local cell.

-fast
: Directs the Cache Manager to display its current list of contacted File Server machines without probing the machines. This option can be combined with the **-cell** or **-all** option; it can also be used if both the **-cell** and **-all** options are omitted.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm statservers** command lists all File Server machines in the indicated cells that meet the following two conditions:

- The Cache Manager has been in contact with the File Exporter running on the machine and needs to contact it in the future (probably because the Cache Manager is holding tokens for data on that File Server machine).
- The File Exporter on the machine is not currently responding to the Cache Manager's probes (implying that it is not responding to the Cache Manager's requests for data either).

The Cache Manager maintains a list of File Server machines that meet the first condition, updating the list periodically by attempting to contact the File Exporter on each machine in the list. When a machine does not respond to a probe, the Cache Manager marks it as non-functioning. If a machine that previously did not respond begins to respond again, the Cache Manager erases the mark. The Cache Manager maintains this information in the kernel of the local machine.

Without the **-fast** option, this command forces the Cache Manager to update its information immediately (rather than waiting the standard interval). The Cache Manager probes the File Exporters on the machines in the specified cells, records those that do not respond, and reports the results. If you include the **-fast** option, the Cache Manager displays the list of non-functioning machines that it has at the time the command is issued; it does not probe the machines again.

By default, the Cache Manager probes machines in the local cell only. If the **-all** option is used, the Cache Manager probes all machines (from all cells) that meet the first condition. If a *cellname* is specified with the **-cell** option, the Cache Manager probes only the machines in that cell.

The execution of this command can be lengthy if a number of machines in the Cache Manager's list are unresponsive when the command is issued. The Cache Manager waits a standard time-out period before concluding that a File Exporter is not responding; this allows for the possibility of slow cross-network communication. If it is important that the command shell prompt return quickly, run this command in the background. It is harmless to interrupt the command (in OS/390 DFS, enter <**Alt-c**> followed by entering the <**c**> key again).

This command does not check the status of all File Server machines in a cell. The Cache Manager probes only those machines that meet the first condition in the previous list.

Privilege Required

No privileges are required.

Output

If the Cache Manager gets a response from all of the machines that it probes (that is, all such machines are functioning normally), the command displays the following output:

```
All servers are running.
```

This message does not imply that all File Server machines in the specified cells are running; it implies only that those machines that the Cache Manager probed are running.

If one or more machines fail to respond to the Cache Manager's probes within the time-out period, the command displays the following output:

```
These servers are still down: hostname.
```

where *hostname* is the name of each File Server machine that fails to respond. In a multihomed server environment (an File Server machine can have up to four IP addresses listed in the Cache Manager's preferencees), the *hostname* corresponds to the host name or IP addresses that the Cache Manager is currently using to access each particular File Server machine. The output does not contain multiple machine names for the same File Server machine.

Examples

The following command uses the **-fast** option to view the Cache Manager's current list of unresponsive machines belonging to the local cell rather than waiting for the Cache Manager to probe them again. The output indicates that all machines responded to the most recent probes.

```
$ cm statservers -f

All servers are running.
```

The following command checks all File Server machines from which the Cache Manager has cached data, regardless of the cell in which a machine resides. The command reports that the machines named **fs1.abc.com** and **fs3.state.edu** did not respond to the Cache Manager's probes.

```
$ cm statservers -all

These servers are still down: fs1.abc.com fs3.state.edu.
```

Related Information

Commands:
cm lsstores **cm whereis**

cm sysname

Purpose

Reports or sets the CPU/OS type.

Format

cm sysname [-newsys *sysname***] [-help]**

Options

-newsys *sysname*
: Specifies the new setting of the CPU/Operating System (**@sys**) variable for the machine on which it is issued. If this option is omitted, the output shows the current setting of the variable.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm sysname** command displays the current setting of the **@sys** variable or sets the variable on a client machine. If the **-newsys** option is omitted, the command reports the current setting of the **@sys** variable. If the **-newsys** option is included, the command sets the variable to the specified CPU/OS type. The value of the variable is displayed from or set in the kernel of the client machine on which the command is issued.

The Cache Manager's main use of the **@sys** variable is in pathnames used in symbolic links. As the Cache Manager interprets pathnames, it substitutes the value of the indicator for any occurrence of **@sys**. (Use the **@sys** variable sparingly; it can make the effect of changing directories confusing.)

Privilege Required

To view the current setting of **@sys** (without the **-newsys** option), no privileges are required. To change the setting of **@sys** (with the **-newsys** option), you must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Output

If the **-newsys** option is not specified, the output reports the system type in the following format:

```
Current sysname is 'system_type'.
```

Examples

The following command shows the output produced on a machine running OS/390 DFS:

```
$ cm sys

Current sysname is 's390_os390'.
```

The following commands set the system type on a machine running AIX 3.2 and use it in a symbolic link from the **/usr/local** directory on the local machine to a directory in the DFS filespace:

```
# cm sys -new rs_aix32
# ln -s /.../abc.com/fs/@sys/usr/local /usr/local
# ls -l /usr/local

lrwxrwxrwx 1 root 34 May 31 1993 /usr/local ->
/.../abc.com/fs/@sys/usr/local

# cd /usr/local
# pwd
/.../abc.com/fs/rs_aix32/usr/local
```

Implementation Specifics

On OS/390 DFS, the default CPU/Operating type (*sysname*) is **s390_os390**.

cm whereis

Purpose

Reports names of File Server machines housing specified files or directories.

Format

cm whereis [-path {*filename* **|** *directory_name***}...] [-help]**

Options

-path {*filename* **|** *directory_name***}**
: Specifies the pathname of each file or directory whose location is to be reported. Each file or directory must reside in DFS, not on a local disk. If a full pathname is not provided, the file or directory is assumed to reside in the current working directory. If this option is omitted, the current working directory is used.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **cm whereis** command displays information about the location of each file or directory indicated with the **-path** option. The command reports the name of the cell in which the file or directory exists, the name of the fileset in which it resides, and the name of each File Server machine that houses a copy of the fileset.

Privilege Required

No privileges are required.

Output

The output includes a separate line displaying the following information about each file or directory specified with the **-path** option:

```
The file 'filename' resides in the cell 'cellname', in fileset
'fileset_name', on hosts hostname, hostname, hostname.
```

In the output,

- *filename* is the pathname of a file or directory specified with the **-path** option.
- *cellname* is the name of the cell in which the file or directory is located.
- *fileset_name* is the name of the fileset where the file or directory is located.
- *hostname* is the name of one or more File Server machines on which the fileset is located. If the fileset is a read/write fileset, only one machine name is displayed; if the fileset is a read-only fileset, multiple machine names can be displayed.

Examples

The following command indicates that the directory **/.../abc.com/fs/bin/sysfile** is located in a replicated fileset on the File Server machines named **fs1**, **fs3**, and **fs6**, all of which are located in the cell named **abc.com**:

```
$ cm whereis /.../abc.com/fs/bin/sysfile

The file '/.../abc.com/fs/bin/sysfile' resides in the cell 'abc.com',
in fileset 'sysfile.bin', on hosts fs1.abc.com, fs3.abc.com, fs6.abc.com.
```

Related Information

Command:
cm statserver

dfsbind

Purpose

Provides user-space information to the Cache Manager and File Exporter.

Important Note to Users

On OS/390 DFS, the **dfsbind** process is handled in an implementation-specific manner. This process runs as part of the **DFSCM** and **dfskern** processes. For further information, see Chapter 13, "Configuring the Cache Manager" on page 251.

The **dfsbind** command is **not** supported on OS/390 DFS. Only the following options are supported as **DFSCM** initialization parameters: **-junctionlife** *seconds_to_live*, **-prefixlife** *seconds_to_live*, and **-notfoundlife** *seconds_to_live*.

Format

dfsbind [-junctionlife *seconds_to_live***] [-prefixlife** *seconds_to_live***] [-notfoundlife** *seconds_to_live***]**

Options

-junctionlife *seconds_to_live*
: Specifies the length of time for which information cached about Fileset Database machines for a cell remains valid. When the Cache Manager (**DFSCM**) retrieves this information from the DFS junction of a cell, it sends the information, along with a "time to live" (TTL). The TTL specifies the length of time for which the Cache Manager is to consider the information valid. The Cache Manager continues to recognize the information as valid until the TTL expires, after which the information is refreshed to refresh the information the next time it is needed.

  By default, **DFSCM** assigns a TTL of 24 hours to information about Fileset Database machines. This option can be used to change the TTL that **DFSCM** assigns to such information. Specify an integer greater than or equal to 30 to indicate the new TTL in seconds.

  Note: This option is only useful for debugging the DFS client.

-prefixlife *seconds_to_live*
: Specifies the length of time for which information cached about a pathname that is a valid DFS junction name prefix remains valid. When the Cache Manager (**DFSCM**) successfully traverses a given path but the path is not a DFS junction name, it caches the valid pathname and TTL. The Cache Manager caches the information and the TTL, continuing to recognize the information as valid until the TTL expires, after which the information is refreshed the next time it is needed.

  By default, **DFSCM** assigns a TTL of 24 hours to information about pathnames that are valid DFS junction name prefixes. This option can be used to change the TTL that **DFSCM** assigns to such information. Specify an integer greater than or equal to 30 to indicate the new TTL in seconds.

  Note: This option is only useful for debugging the DFS client.

-notfoundlife *seconds_to_live*
: Specifies the length of time for which information cached about an invalid pathname remains valid. When the Cache Manager (**DFSCM**) cannot traverse a given path, it caches the pathname and TTL. The Cache Manager considers the cached information valid until the TTL expires. After the TTL expires, the Cache Manager refreshes the information the next time it is needed.

By default, **DFSCM** assigns a TTL of one hour to information about invalid pathnames. This option can be used to change the TTL that **DFSCM** assigns to such information. Specify an integer greater than or equal to 30 to indicate the new TTL in seconds.

Note: This option is only useful for debugging the DFS client.

Usage

On a client machine, **dfsbind** performs the following services:

- It contacts CDS to resolve DCE pathnames (both local and foreign) that it receives from the Cache Manager. When a user on a client machine requests data that the Cache Manager has not cached, the Cache Manager employs **dfsbind** to resolve the pathname of the data. It sends **dfsbind** each element of the pathname in succession, appending each new element to the preceding elements when it sends it (for example, it first sends */.../element_one*, then */.../element_one/element_two*, and so on). In turn, **dfsbind** determines whether each successive pathname is valid.

  If the pathname is valid, it eventually contains a DFS junction from which **dfsbind** can access information about the Fileset Database machines for the cell in which the data resides. If it encounters a junction for the DFS filespace, **dfsbind** returns information about the names and network addresses of the Fileset Database machines for the cell to the Cache Manager. (It actually decomposes binding handles to learn this information.) The Cache Manager uses the information from **dfsbind** to create an RPC binding that it employs to communicate with a Fileset Location (FL) Server on an appropriate Fileset Database machine. The **flserver** examines the FLDB and tells the Cache Manager which File Server machine houses the fileset that contains the data requested by the user.

  For each successive pathname that it attempts to resolve for the Cache Manager, **dfsbind** returns one of the following error codes to the Cache Manager to indicate the result of the resolution operation:

`0`	Indicates that the pathname corresponds to a DFS junction that contains information about the Fileset Database machines in the cell. The process sends information about the Fileset Database machines to the Cache Manager.
`EISDIR`	Indicates that the pathname is a valid DFS junction name prefix but is not itself a DFS junction. The process returns the valid pathname to the Cache Manager.
`ENOENT`	Indicates that the given path could not be traversed. The process returns the invalid pathname to the Cache Manager.
`ETIMEDOUT`	Indicates that unexpected errors occurred. The process returns only the error code to the Cache Manager.

  DCE pathname and DFS junction information that the Cache Manager receives from **dfsbind** is valid for a limited amount of time. **dfsbind** associates a "time to live" (TTL) with all information it sends to the Cache Manager. The TTL defines the amount of time for which the Cache Manager is to consider the information valid. The Cache Manager caches the TTL with the information. Once its TTL has elapsed, the information becomes stale; the Cache Manager contacts **dfsbind** to refresh the information the next time it needs it.

  dfsbind associates the following TTLs with the information it passes to the Cache Manager:

  – Information about Fileset Database machines (error code 0) receives a TTL of 24 hours by default. (The TTL of such information can be modified with the command's **-junctionlife** option.)

  – Information about valid DFS junction name prefixes (error code EISDIR) has a TTL of 24 hours by default. (The TTL of this type of information can be changed with the command's **-prefixlife** option.)

  – Information about invalid pathnames (error code ENOENT) has a TTL of 1 hour by default. (The TTL of this type of information can be altered with the command's **-notfoundlife** option.)

For example, when the Cache Manager first needs to access data from a fileset in the local cell, it passes each successive element of the DCE pathname of the data to **dfsbind**. If the path contains a DFS junction name, **dfsbind** eventually returns information about the local cell's Fileset Database machines, and a TTL that it assigns to the information, to the Cache Manager. The Cache Manager caches the information and the TTL, using the information to contact a Fileset Database machine in the cell. If the Cache Manager needs to access data from a fileset in the local cell before the TTL has elapsed, it uses the cached information to contact a Fileset Database machine in the cell. However, if it needs to access data from a fileset in the local cell after the TTL has elapsed, it again contacts **dfsbind** to refresh its knowledge of local Fileset Database machines.

- It obtains user authentication information for the RPC Runtime. It communicates with the Security Service of the appropriate cell to obtain authentication information about users of the client machine.

  The Cache Manager communicates with the RPC Runtime when it needs to create an RPC binding to a File Server machine on behalf of a user. The RPC Runtime then communicates with **dfsbind** to obtain authentication information about the user for use in the binding. **dfsbind** obtains the authentication information from the Security Server and sends it back to the RPC Runtime, which packages the information along with the other information from the Cache Manager into the RPC binding and sends it to the appropriate File Server machine.

On a File Server machine, **dfsbind** simply maintains user authentication information required by the File Exporter on the machine. The File Exporter uses this information to ensure that only authenticated users access data from the machine.

Implementation Specifics

On OS/390 DFS, the **dfsbind** process is handled in an implementation-specific manner. This process runs as part of the **DFSCM** and **dfskern** processes. For further information, see Chapter 13, “Configuring the Cache Manager” on page 251.

The **dfsbind** command is **not** supported on OS/390 DFS. Only the following options are supported as **DFSCM** initialization parameters: **-junctionlife** *seconds_to_live*, **-prefixlife** *seconds_to_live*, and **-notfoundlife** *seconds_to_live*.

Related Information

Commands:

dfsd	**fxd**

dfsd

Purpose

Initializes the DFS Cache Manager and starts related daemons.

Important Note to Users

On OS/390 DFS, the **dfsd** process is handled in an implementation-specific manner. This process runs as part of the **DFSCM** process.

The **dfsd** command is **not** supported on OS/390 DFS. The options described for the **dfsd** command can be specified as **DFSCM** initialization parameters. For further information, see Chapter 13, "Configuring the Cache Manager" on page 251 and "Cache Manager Initialization Parameters" on page 273.

In addition, three implementation-specific options are provided on OS/390 DFS: **-translation** *translation_options*, **-mountfilesystem** *virtual_HFS_file_system_name*, and **-log {on | off}**.

Format

dfsd [-blocks *number_of_cache_blocks***]**
[-files *number_of_cache_files***]**
[-stat *number_of_status_cache_entries***]**
[-rootfileset *root_fileset***]**
[-cachedir *minor_device_number***]**
[-mountdir *DFS_mount_directory***]**
[-rootcell *root_cell***]**
[-mainprocs *number_of_background_daemons***]**
[-tokenprocs *number_of_token_daemons***]**
[-ioprocs *number_of_I/O_background_daemons***]**
[-memcache] [-dcache *number_of_entries***]**
[-chunksize *chunk_exponent***]**
[-namecachesize *number_of_name_cache_entries***]**
[-initiallocalprotectlevel *level***]**
[-minlocalprotectlevel *level***]**
[-minremoteprotectlevel *level***]**
[-initialremoteprotectlevel *level***]**
[-verbose] [-debug]
[-callbackhint *callback_IP_address_hint***]**
[-translation {<u>binary</u> | text | auto}]
[-mountfilesystem *virtual_HFS_file_system_name***]**
[-log {<u>on</u> | off}]

Options

-blocks *number_of_cache_blocks*
Specifies the number of kilobytes to be made available for caching in the machine's cache directory (for a disk cache) or memory (for a memory cache). This value overrides the default, which must be specified in the third field of the **/opt/dcelocal/etc/CacheInfo** file. The unit of measure for block size is always kilobytes.

A disk cache should not exceed 75% of the disk space available on the cache partition. This limit is necessary because the implementation of the cache requires a small amount of disk space.

For a memory cache, do not combine this option with the **-dcache** option.

Note: The minimum cache size you can specify with the **-blocks** option is 17 kilobytes. If you specify a cache size smaller than 17 kilobytes, the Cache Manager creates a cache of 17 kilobytes.

-files *number_of_cache_files*
Specifies the number of V files (chunks) to be created in the cache directory for a disk cache. This value overrides the default, which is the number of cache blocks divided by 8.

Each V file can accommodate a chunk of data. By default, each chunk can accommodate 64 kilobytes of data. To operate most efficiently, at least 90 percent of the cache must be in use. Use the **-files** option to increase the number of V files if this is not the case. Do not specify a value greater than 32,000.

Do not combine this option with the **-memcache** option, which is used for memory caching.

Note: The minimum number of V files you can specify with the **-files** option is 2. If you specify a value smaller than 2, the Cache Manager creates a cache with two V files.

-stat *number_of_status_cache_entries*
Specifies the number of entries in the machine's memory for recording status information about DFS files in the cache. The default is 300.

-rootfileset *root_fileset*
Names the read-write fileset corresponding to the top-level (**root**) directory. Generally used for testing purposes only.

-cachedir *minor_device_number*
Specifies the *minor_device_number* of the local DCE Local File File System aggregate to be used for DFS client disk caching. This value overrides the default, which must be specified in the second field of the **CacheInfo** file.

Do not combine this option with the **-memcache** option, which is used for memory caching. With memory caching, the **-cachedir** option, like the second field of the **CacheInfo** file, is ignored.

-mountdir *DFS_mount_directory*
Names the local disk directory where the DCE global namespace is to be mounted. This overrides the default, which must be specified in the first field of the **CacheInfo** file. The default for a machine with a disk is the global namespace designation (**/...**); if **/...** is not used, any symbolic links to the global namespace will not work.

Note: On OS/390 DFS, this option should not be used.

-rootcell *root_cell*
Names the cell that contains the root fileset.

Note: On OS/390 DFS, this option should not be used.

-mainprocs *number_of_background_daemons*
Specifies the number of background daemons to run on the machine. These daemons improve efficiency by performing prefetching and background writing of saved data. The default is two.

Increase the number of background daemons if the machine serves more than five users.

-tokenprocs *number_of_token_daemons*
: Specifies the number of background daemons dedicated to servicing token revocation RPC requests from File Exporters. The default is two. (Token daemons run in addition to the background daemons associated with the **-mainprocs** option.)

Increase the number of token daemons if users on this machine interact with many File Server machines.

-ioprocs *number_of_I/O_background_daemons*
: Specifies the number of background I/O daemons performing I/O operations. The default is five.

-memcache
: Causes **DFSCM** to initialize a memory cache rather than a disk cache. If this option is provided, space in memory is allocated for the cache; no disk space is used, even if it is available.

Do not combine this option with the **-files** option (which is used for machines that use disk caching). Also, do not combine this option with the **-cachedir** option; with memory caching, the **-cachedir** option, like the second field of the **CacheInfo** file, is ignored.

-dcache *number_of_entries*
: Sets the number of dcache entries in memory; dcache entries store information about cache chunks.

For a disk cache, the **/opt/dfslocal/var/adm/dfs/cache/CacheItems** file contains one entry for each V file. By default, 100 entries from the **CacheItems** file are duplicated in machine memory; the **-dcache** option overrides the default.

For a memory cache, there is no **CacheItems** file; one dcache entry exists for each cache chunk. The Cache Manager determines the number of dcache entries (cache chunks) by dividing the cache size by the chunk size; the **-dcache** option sets the number of cache chunks. Do not combine this option with the **-blocks** option.

Use of this option with a disk cache is not necessary because it increases performance only marginally. It is not recommended with a memory cache because it requires the issuer to perform additional calculations.

-chunksize *chunk_exponent*
: Sets the size of each cache chunk. Provide an integer between 13 and 18 to be used as an exponent of 2. This value overrides the default chunk size, which is 64 kilobytes (2^{16}) for a disk cache and 8 kilobytes (2^{13}) for a memory cache. A value less than 13 or greater than 18 sets the chunk size to the appropriate default for the type of cache in use. The unit of measure for chunk size is always bytes.

It is not recommended that you use this option with the **-dcache** option for a memory cache.

-namecachesize *number_of_name_cache_entries*
: Sets the number of entries allocated for the Cache Manager's name lookup cache. Provide an integer greater than 0 (zero); the default number of name cache entries is 256.

The name lookup cache stores the results obtained from remote directory lookup requests to DFS servers, which allows subsequent lookup requests for the same file or directory to be satisfied on the local DFS client rather than on the remote DFS server. Because name cache entries are recycled when the name lookup cache limit is reached, the ability to satisfy the request locally depends on the size of the name lookup cache.

-initiallocalprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value

between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string.

-minlocalprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within the same cell. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string.

-initialremoteprotectlevel *level*
: Specifies the initial DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string.

-minremoteprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the Cache Manager and File Servers within foreign cells. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string.

-verbose Directs **DFSCM** to produce a more detailed trace of its activities than it does by default. The trace is displayed on standard output (**STDOUT**) unless it is directed elsewhere.

-debug Causes **DFSCM** to produce a highly detailed trace of its activities, which can be useful for debugging purposes. The trace is displayed on standard output (**STDOUT**) unless it is directed elsewhere.

-callbackhint *callback_IP_address_hint*
: Specifies an IP address hint to be considered by the DFS client when selecting the callback address to be provided to the DFS fileserver during connection setup. The IP address is specified using dot notation (for example, **130.12.45.2**). Normally, the DFS client will attempt to choose the best address when there is more than one to choose from. The correct choice may not be made in all circumstances. The **callbackhint** provides a mechanism to bias the selection process.

  This option is useful in environments where the DFS client has multiple configured network interfaces and full connectivity does not exist between the DFS fileserver and the DFS client over all of the configured interfaces.

-translation <u>binary</u> | text | auto
: Defines the type of character translation. This is an implementation-specific option of OS/390 DFS. If **binary** is specified, file data received (read) or sent (written) by the DFS client is not translated. The default is **binary**.

  If **text** is specified, file data received (read) or sent (written) by the DFS client is translated. After reading file data, the file data is assumed to be in ISO 8859-1 (ASCII) and is translated to the local code page with the default translation tables. Before writing file data, the file data is translated to the ISO 8859-1 (ASCII) code page from the local code page with the default translation tables.

  If **auto** is specified, file data received (read) or sent (written) by the DFS client is examined to determine if the data is text or not. When data is received (read), the DFS client compares the first 255 bytes of data against a table of valid text (ASCII) characters. If all 255 characters are deemed to be valid text characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated. When data is sent (written), the DFS client compares the first 255 bytes of data against a table of valid text (EBCDIC) characters. If all 255 characters are deemed to be valid text characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated.

Note: If a **cmattr** file is specified, and the file name suffix matches a directive in the **cmattr** file, that controls whether translation occurs. The **-translation** specification is used when no **cmattr** file is specified, or the file name suffix does not match any of the directives in the **cmattr** file, or the file name has no suffix.

-mountfilesystem *virtual_HFS_file_system_name*
Specifies the name of a virtual HFS file system that the DFS Client (**DFSCM**) mounts on the **/...** directory. The HFS file system need not be allocated but the specified file system name should be unique. This is an implementation-specific option of OS/390 DFS.

It is recommended that the name **IOE_DFS_CLIENT_DATA** be used to uniquely identify the DFS client to UNIX System Services. This name is supplied by IBM and included in the **DFSCM** environment variable file.

Note: A **-mountfilesystem** name must be specified as an **_IOE_CM_PARMS** value in the **DFSCM** environment variable file (for further information, see “DFS Client ENVAR File” on page 274).

-log <u>**on**</u> **|** **off**
Specifies whether messages from the OS/390 DFS client are written to the file **/opt/dcelocal/var/dfs/adm/CMLog**. The log file may be viewed using the **bos getlog** command.

Usage

The **dfsd** process initializes the DFS Cache Manager on a client machine according to the information specified with the options described previously. It must be run on all DFS client machines. It is usually added to the proper start-up file (**/etc/rc** or its equivalent) rather than typed at the command shell prompt. (The **dfsbind** process must be run before the **dfsd** process in a start-up file.) The binary file for the **dfsd** process resides in *dcelocal***/bin/dfsd**.

Specifically, the **dfsd** process does the following:

- Determines if the cache is on the local disk or in machine memory. A disk cache is used unless the **-memcache** option is provided. If the **-memcache** option is used, no disk space is used, even if it is available; the Cache Manager maintains all cached data and cache-related information in memory.
- Defines the *minor_device_number* of the local DCE Local File System aggregate to be used for DFS client disk caching. This value overrides the default which must be specified in the second field of the **CacheInfo** file.
- Sets the size of the cache. The third field in the **CacheInfo** file specifies the default cache size in kilobytes.

  For a disk cache, the value in the **CacheInfo** file is an upper limit that can be increased only with the **-blocks** option; it cannot be increased with the other options available with the **dfsd** process. For a memory cache, the **-dcache** option alone or in combination with the **-chunksize** option overrides the cache size specified in the **CacheInfo** file; these combinations are not recommended.

  After initialization, use the **cm setcachesize** command to change the size of a disk cache without rebooting. The value set with the **cm setcachesize** command is overridden the next time the machine is rebooted and **dfsd** is run. The **cm setcachesize** command does not work for memory caches; the machine must be rebooted. (The **cm getcachesize** command can be used to display the current size of the cache, the amount in use, and the type of cache—disk or memory.)
- Sets the size of each chunk of data in the cache and, by implication, the amount of data the Cache Manager requests at one time from the File Exporter. For a memory cache, if the total cache size divided by the chunk size leaves a remainder, **dfsd** rounds the number down, resulting in a slightly smaller cache.

- Sets the number of dcache entries allocated in machine memory for storing information about the cache chunks in a disk cache.
- Sets the number of empty V files created in the cache directory for a disk cache. (A memory cache cannot use V files because it does not use disk storage; the number of chunks is instead equal to the number of dcache entries.)
- Sets the number of **stat** entries in machine memory for caching status information about cached DFS files.
- Sets the number of entries in the name lookup cache for storing the results of remote directory lookups.
- Specifies the directory on the machine's local disk where DFS is mounted. The first field in the **CacheInfo** file specifies the default directory.
- Sets the initial RPC authentication level and minimum RPC authentication bound for communications between the Cache Manager and File Servers.

In addition to setting cache configuration parameters, **dfsd** also starts the following types of daemons. On most system types, these daemons appear as nameless entries in the output of the **ps** command.

- One or more maintenance daemons, which perform routine periodic maintenance tasks such as the following:
  - Performing garbage collection
  - Synchronizing files
  - Probing processes on File Server machines every few minutes
  - Refreshing information about filesets referenced by the Cache Manager once per hour
- One or more background daemons, which improve performance by performing delayed writing of updated data. The default number of background daemons is two, which is usually sufficient to handle up to five simultaneous users of a machine. Use the **-mainprocs** option to increase the number of background daemons if the machine serves more than five users.
- One or more token daemons, which handle token revocation RPC requests from the File Exporters on File Server machines (for example, by writing modified data back to the File Server machines). The default number of token daemons is two. Use the **-tokenprocs** option to increase this number if the machine interacts with many File Server machines from different cells.

The default number of daemons is ten (one maintenance daemon, two background daemons, two token daemons, and five I/O daemons). You can alter only the number of background daemons, token daemons, and I/O daemons; **dfsd** initializes additional maintenance daemons as necessary.

RPC Security Settings: The **dfsd** command sets the DCE RPC security level for RPCs sent between a Cache Manager and DFS File Servers. The command sets two levels: an initial DCE RPC security level used as a starting point in security level negotiations between the Cache Manager and a File Server, and the minimum DCE RPC security level that the Cache Manager will accept for such communications. Two sets of these levels are maintained: One set specifies the security levels for communications with File Servers within the local cell, and the other set specifies the security levels for communications with File Servers within foreign cells. Both sets of security levels can be adjusted through the **cm setprotectlevels** command.

In operation, the Cache Manager and File Server interact to arrive at a mutually acceptable authentication level for communications. The negotiation starts with an RPC that uses the initial authentication level sent from the Cache Manager to the File Server. If the initial authentication level is outside the minimum or maximum bounds set at the File Server, the File Server returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Server requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Server). Once the Cache Manager and File Server have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Server.

The Cache Manager and **fxd** default settings are such that communications occur at the packet authentication level for local cell and packet integrity authentication level for non-local cells.

In addition to a general pair of upper and lower bounds for all communications between the File Server and Cache Manager, administrators can also set advisory bounds on a per-fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level (they may be enforced in a future version of DFS). Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset.

The various authentication levels are set by specifying either an integer value between 0 and 7, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **rpc_c_protect_level_default** or **default** or **0**: Use the DCE default authentication level.
- **rpc_c_protect_level_none** or **none** or **1**: Perform no authentication.
- **rpc_c_protect_level_connect** or **connect** or **2**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_c_protect_level_call** or **call** or **3**: Authenticate only at the beginning of each RPC received.
- **rpc_c_protect_level_pkt** or **pkt** or **4**: Ensure that all data received is from the expected host.
- **rpc_c_protect_level_pkt_integrity** or **pkt_integrity** or **5**: Authenticate and verify that none of the data transferred has been modified.
- **rpc_c_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.
- **rpc_c_protect_level_cdmf_priv** or **cdmf_priv** or **7**: Perform authentication as specified by all of the previous levels (except 6) and also encrypt each RPC argument value. This level provides a lower level of packet privacy than **rpc_c_protect_level_pkt_privacy**.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

Privilege Required

The issuer must be **root** on the local machine.

Implementation Specifics

On OS/390 DFS, the **dfsd** process is handled in an implementation-specific manner. This process runs as part of the **DFSCM** process. For further information, see Chapter 13, “Configuring the Cache Manager” on page 251.

The **dfsd** command is **not** supported on OS/390 DFS. In addition, three implementation-specific options are provided on OS/390 DFS: **-translation** *translation_options*, **-mountfilesystem** *virtual_HFS_file_system_name*, and **-log {on | off}**.

Related Information

Commands:

cm getcachesize	**cm setcachesize**	**dfsbind**
cm getprotectlevels	**cm setprotectlevels**	**fts setprotectlevels**

Files:

CacheInfo	**FilesetItems**	**Vn**
CacheItems		

dfsexport

Purpose

Exports (or unexports) DCE Local File System aggregates and non-Local File System partitions to the DCE namespace.

Format

dfsexport [{-all | -aggregate *name***}] [-type** *name***] [-detach] [-force] [-verbose] [-help]**

Options

-all
Specifies that all aggregates and partitions listed in the **/opt/dcelocal/var/dfs/dfstab** file are to be exported. Use the **-type** option with this option to export only DCE Local File System aggregates or only non-Local File System partitions. Use this option or use **-aggregate**; omit both options to list all aggregates and partitions currently exported from the local disk to the DCE namespace.

-aggregate *name*
Specifies the device name or aggregate name of the aggregate or partition to be exported. These names are specified in the first and second fields of the entry for the aggregate or partition in the **dfstab** file. Use this option or use **-all**; omit both options to list all aggregates and partitions currently exported from the local disk to the DCE namespace.

Note: If you enter this command in TSO/E under OS/390 DFS, the *name* parameter **must** be surrounded by double quotes ("").

-type *name*
Specifies that only aggregates or partitions whose file system types match the type specified with this option are to be exported. The type can be specified as **lfs** to export only DCE Local File System aggregates, or it can be specified as **ufs** to export only non-Local File System partitions. The type of each aggregate or partition appears in the third field of the entry for the device in the **dfstab** file. The type must be specified in lowercase letters (as it appears in the **dfstab** file).

Use this option only with the **-all** option; it is ignored if it is used without the **-all** option. If it is omitted and **-all** is used, the command exports both **lfs** and **ufs** devices.

-detach
Used with the **-all** option, specifies that the aggregates or partitions indicated with the command's other options are to be detached (no longer exported), making them unavailable by way of the DCE namespace. Use **-all** or **-aggregate** with this option to indicate the devices to be detached; use the **-type** option with **-all** to detach only one type of device.

Use the **-detach** option only when no users are accessing data on the aggregate or partition to be detached or when a serious emergency warrants its use. When the **-detach** option is used, the command revokes all tokens for data on a device before it detaches it. It does not detach a device unless it can revoke all necessary tokens. You can use the **-force** option to direct the command to detach a device even if it cannot revoke all necessary tokens.

To permanently detach an aggregate or partition, it must also be removed from the **dfstab** file. Otherwise, the **dfsexport** command exports the aggregate or partition the next time it is run (provided the aggregate or partition is included in the specification for the devices to be exported).

-force	Used with the **-detach** option, directs the **dfsexport** command to detach an aggregate or partition even if it cannot revoke all tokens for data on the device. By default, the command does not detach a device unless it can revoke all necessary tokens. Use this option only when a serious emergency requires its use. This option can be used only with the **-detach** option. The command is not successful if this option is used with any combination of options that does not include the **-detach** option. Use this option only when no users are accessing data on the aggregates or partitions to be detached or when a serious emergency requires its use.
-verbose	Directs the command to report on its actions as it executes.
-help	Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **dfsexport** command exports DCE Local File System aggregates and non-Local File System disk partitions from the local disk of a machine to the DCE namespace. File systems on exported aggregates and partitions are available to other users in the DCE namespace. Issue this command with no options to list the aggregates and partitions already exported. The binary file for the **dfsexport** command resides in **/opt/dfsglobal/bin/dfsexport**.

The **dfsexport** command exports DCE Local File System aggregates, non-Local File System partitions, or both based on the values provided with its options. If the **-all** option is provided, the command exports all aggregates and partitions listed in the **/opt/dcelocal/var/dfs/dfstab** file. If the **-aggregate** option is provided, it exports only the aggregate or partition whose device name or aggregate name is specified with the option. The specified name must be listed in the **dfstab** file.

The **-type** option can be used with the **-all** option to indicate that only DCE Local File System aggregates or only non-Local File System partitions are to be exported. If **lfs** is provided with the **-type** option, the command exports only DCE Local File System aggregates; if **ufs** is provided with the **-type** option, it exports only non-Local File System partitions. If the **-type** option is used, the **-all** option must also be included; otherwise, the **-type** option is ignored.

When **dfsexport** executes, it reads the **dfstab** file on the local disk of the machine to determine the aggregates and partitions available to be exported. An aggregate or partition must have an entry in the dfstab file if it is to be exported. Because this command reads the **dfstab** file, information supplied with its options must match exactly the information for an aggregate or partition specified in that file.

The **dfsexport** command reads a list of all currently exported aggregates and partitions that is maintained in the DFS Server of the local machine. The command will not export an aggregate or partition that is currently exported. The command also refuses to export a DCE Local File System aggregate that needs to be recovered with the **salvage** command. If this is the case, use the **salvage** command to recover the aggregate that caused the failure and reissue the **dfsexport** command.

Issuing the **dfsexport** command with no options lists the aggregates and partitions currently exported from the local file server to the DCE namespace. The **fts lsaggr** command can also be used to display a current list of all aggregates and partitions exported from a machine.

The **dfsexport** command is generally included in a machine's initialization file (**/etc/rc** or its equivalent) rather than issued at the keyboard. On OS/390 DFS, this is controlled by the **export** entry of the **ioepdcf** file, see “ioepdcf” on page 382 for more information. Once included in the initialization file, the command automatically exports all indicated aggregates and partitions listed in the **dfstab** file whenever the machine is rebooted. Typically, the command is included with its **-all** option to export all aggregates and partitions listed in the **dfstab** file.

Prior to using this command to export a non-Local File System partition for the first time, perform the following (see "Exporting Non-Local File System Partitions" on page 171 for more information):

1. Ensure that the partition is mounted locally; it can contain data or it can be empty. To export OS/390 Data Sets (referred to as Record File Systems (RFS)), local mounting is not required or supported.
2. Issue the **fts crfldbentry** command to register the non-Local File System fileset that resides on the partition (each non-Local File System partition contains a single fileset) in the Fileset Location Database (FLDB). The Fileset Location Server (**flserver**) can then track the fileset's location. The **fts crfldbentry** command also allocates a unique fileset ID number for the non-Local File System fileset.
3. Create an entry for the non-Local File System partition in the **devtab** file on the machine on which the partition resides. This entry maps the minor device number to the HFS data set or RFS data set (prefix) you wish to export. It also allows you to (optionally) specify whether character data translation should occur when the data is read or written by DFS clients. For further information, see "devtab" on page 368.
4. Create an entry for the non-Local File System partition in the **dfstab** on the machine on which the partition resides. Use the aggregate ID number specified with the **-aggrid** option of the **fts crfldbentry** command and the fileset ID number generated by the command in the appropriate fields of the entry for the partition.

   Also, use the name of the partition's local mount point as its aggregate name in the second field of its entry. (Once these steps are complete, use the **fts crmount** command to mount the non-Local File System fileset that resides on the partition.)

Before exporting a non-Local File System partition, also make sure that no users have files open on the partition. DFS cannot effectively synchronize file access between users who opened files from a non-Local File System partition before the partition was exported and users who open files from the partition after the partition is exported because only the latter have tokens.

Before using the command to export a DCE Local File System aggregate for the first time, complete the following steps (see "Exporting DCE Local File System Aggregates" on page 166 for more information):

1. Create an entry for the non-Local File System partition in the **devtab** file on the machine on which the partition resides. This entry maps the minor device number to the DCE Local File System aggregate you wish to export. For further information, see "devtab" on page 368.
2. Create an entry for the DCE Local File System aggregate in the **dfstab** file on the machine on which the aggregate is located. (Once the aggregate is exported, the **fts create** command can be used to create and register filesets on the aggregate, after which the **fts crmount** command must be used to mount the new filesets.)
3. Ensure that the disk partition where the aggregate is to reside is initialized with the **newaggr** command and it must not contain any data. The **newaggr** command needs to be run on a partition only once. Do not use the **newaggr** command to reinitialize a partition that contains data you want to preserve; the command destroys any data on the partition on which it is used.

The **dfsexport** command can also be used to detach an exported aggregate or partition from the DCE namespace. Detaching an aggregate or partition makes it unavailable in the namespace. To detach one or more aggregates or partitions, use the **-all** (and optionally the **-type**) option or the **-aggregate** option to specify the devices to be detached, and include the **-detach** option with the command.

Before it detaches a device, the command revokes all tokens for data on the device. When their tokens are revoked, clients flush data cached from the device, writing any modified data back to the device. If the command cannot revoke all necessary tokens, it does not detach the device (it instead displays a message reporting that the device is busy). The **-force** option can be used with the **-detach** option to direct the command to detach a device even if it cannot revoke all necessary tokens (that is, even if files from the device are still open).

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Cautions

Before using the **-detach** option with this command, make sure no users are currently accessing data from filesets on the aggregates or partitions to be detached. The command does not verify that a device is not in use before removing it from the namespace. A user who is accessing data housed on an aggregate or partition when it is detached will not be able to save the data back to the device. Any attempt to perform an action that involves a detached aggregate or partition elicits a message reporting that the device is unknown.

Before detaching an aggregate or partition, attempt to ensure that no users are currently accessing data from filesets on the device. The command revokes all tokens for data on the device before it detaches it, which causes clients to flush data cached from the device (writing any modified data back to the File Server machine). However, a user who is accessing data from the device will no longer be able to save the data. Any attempt to perform an action that involves a detached aggregate or partition elicits a message reporting that the device is unknown. Exercise special caution before using both the **-detach** and **-force** options, which forces a device to be detached even if all tokens cannot be revoked (that is, even if files are still open.)

Examples

The following command line is typically added to a machine's initialization file (**/etc/rc** or its equivalent). The line exports all of the aggregates and partitions that have entries in the machine's **dfstab** file.

```
dfsexport -all
```

On OS/390 DFS, the **dfsexport** process, by default, is started automatically by the DFS control task program, **DFSCNTL**. **DFSCNTL** and its **child** processes are, in turn, controlled in OS/390 DFS by the OS/390 system command, **MODIFY**.

You can also use the **MODIFY** OS/390 system command to export and unexport aggregates on OS/390 DFS.

The following command causes the **dfsexport** program to run with the parameters specified in the **ioepdcf** configuration file. By default, this command exports all of the aggregates and partitions that have entries in the machine's **dfstab** file:

```
modify dfs,start export
```

The following command causes the **dfsexport** program to run with the parameters specified in the **ioepdcf** configuration file. In this example, the command unexports all of the aggregates and partitions that have entries in the **dfstab** file:

```
modify dfs,start unexport
```

The following command exports the aggregate whose device name (as it appears in the **dfstab** file) is **/dev/ufs1**:

```
# dfsexport /dev/ufs1
```

The command that follows exports all DCE Local File System aggregates (all entries in the **dfstab** file with file system type **lfs**):

```
# dfsexport -all -type lfs
```

Implementation Specifics

The **dfsexport** process on OS/390 DFS, by default, is started automatically by the the DFS control task program, **DFSCNTL**. The **dfsexport** process on OS/390 DFS runs under the control of the DFS control task, **DFSCNTL**. **DFSCNTL** and its **child** processes are are controlled on OS/390 DFS by the OS/390 system command **MODIFY**. For further information, see “modify dfs daemon” on page 329.

This command may be issued from TSO/E or an OS/390 shell.

Related Information

Commands:

fts create	**fts crmount**	**newaggr**
fts crfldbentry	**fts lsaggr**	**salvage**

Files:

devtab	**dfstab**

dfsshare

Purpose

An OMVS command that shares (or unshares) HFS directories or printers with SMB clients.

Important Note to Users

This file is for SMB support only.

Format

dfsshare [{-all | -share *name***}] [-type** *name***] [-detach] [-verbose] [-help]**

Options

-all
Specifies that all HFS directories and printers listed in the **/opt/dfslocal/var/dfs/smbtab** file are to be shared with SMB clients. Use the **-type** option with this option to export only HFS files or only OS/390 Infoprint Server printers. Use this option or use **-share**; omit both options to list all file shares and printers currently shared with SMB clients.

-share *name*
Specifies the share name of the HFS directory or printer to be shared. This name is specified in the second field of the entry for the HFS directory or printer in the **smbtab** file. Use this option or use **-all**; omit both options to list all HFS directories and printers currently shared with Windows clients.

-type *name*
Specifies that only directory or printer shares whose type matches the type specified with this option are to be shared. The type can be specified as ufs to share only HFS directories, or it can be specified as prt to share only printers. The type of each share appears in the third field of the entry for the device in the **smbtab file** Use this option only with the **-all** option; it is ignored if it is used without the **-all** option. If it is omitted and **-all** is used, the command shares both ufs and prt types.

-detach
Used with the **-all** option, specifies that the directory and printer shares indicated with the command's other options are to be detached (no longer shared), making them unavailable to SMB clients. Use **-all** or **-share** with this option to indicate the shares to be detached; use the **-type** option with **-all** to detach only one type of share. Use the **-detach** option only when no users are accessing data on the HFS directory shares to be detached or when a serious emergency warrants its use. To permanently detach a share, it must also be removed from the smbtab file. Otherwise, the dfsshare command shares the directories and printers the next time it is run (provided the directories or printers are included in the specification for the types to be shared).

-verbose
Directs the command to report on its actions as it executes.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **dfsshare** command shares HFS directories and printers from OS/390 to Windows clients. Directories and printers that are shared are still available to other OS/390 users. Issue this command with no options to list the directories and printers already shared. The binary file for the dfsshare command resides in **/opt/dfsglobal/bin/dfsshare**.

The **dfsshare** command shares HFS directories, printers, or both based on the values provided with its options. If the **-all** option is provided, the command shares all directories and printers listed in the **/opt/dfslocal/var/dfs/smbtab** file. If the **-share** option is provided, it shares only the directory or printer whose share name is specified with the option. The specified name must be listed in the **smbtab** file.

The **-type** option can be used with the **-all** option to indicate that only directories or only printers are to be shared. If ufs is provided with the **-type** option, the command exports only directories; if prt is provided with the **-type** option, it exports only printers. If the **-type** option is used, the **-all** option must also be included; otherwise, the **-type** option is ignored.

When **dfsshare** executes, it reads the **smbtab** file to determine the directories and printers available to be shared. A directory or printer must have an entry in the **smbtab** file if it is to be shared. This command also invokes **dfsexport** to ensure that the underlying HFS file system that contains the shared directory is exported. Therefore, the corresponding HFS file system information must exist in **dfstab** and **devtab** for the directory shares that are being creaed. If there are additional file systems mounted below the shared directory that you want to allow PC users to access, those file systems must be exported also. The **dfsexport** command can be used for this purpose.

The **dfsshare** command reads a list of all currently shared directories and printers that is maintained in the SMB Server of the local machine. The command does not share a directory or printer that is currently shared. If you want to change the parameters for a printer or HFS directory that is already shared, you must first detach (using the **dfsshare -share** *name* **-detach** command) and then reshare (using the **dfsshare -share** *name* command) for the new parameters to take effect. Issuing the **dfsshare** command with no options lists the directories and printers currently shared to SMB clients.

The **dfsshare** command is automatically executed when the OS/390 SMB server is started (with SMB processing enabled). This is controlled by the export entry of the **ioepdcf** file, see “ioepdcf” on page 382 for more information. When export is enabled, all indicated HFS directories listed in the smbtab file are shared and all HFS file systems listed in dfstab and devtab are exported. In addition, all printers listed in the smbtab are shared (if the OS/390 Infoprint Server is enabled) regardless of the **export** entry in the **ioepdcf** file.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390, **root** refers to a user with a **UID = 0**.

Implementation Specifics

The **dfsshare** command can only be issued from OMVS. It is not supported as a TSO command.

Related Information

Files:

devtab	**dfstab**	**smbtab**

Command:

| ## dfsexport

flserver

Purpose

Initializes the Fileset Location Server (**flserver**).

Format

flserver [-adminlist *filename*] **[-verbose] [-help]**

Options

-adminlist *filename*
Specifies the administrative list file that contains principals and groups authorized to execute **flserver** RPCs (usually using **fts** commands). If this option is omitted, the **flserver** uses the default administrative list file, **/opt/dcelocal/var/dfs/admin.fl**.

-verbose Directs the **flserver** process to provide a detailed report on what it is doing by displaying messages on standard error.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with **flserver**. See "bos help" on page 504 and "bos apropos" on page 484 for examples of using these commands.

Important Note to Users

On OS/390 DFS, the **flserver** process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **flserver**. For further information about configuring the **flserver**, refer to "bos create" on page 485.

Usage

The Fileset Location Server (**flserver**) maintains the Fileset Location Database (FLDB), which tracks the location of all DCE Local File System and non-Local File System filesets. The Fileset Location Server, or **flserver** process, must run on all Fileset Database machines. It is started and controlled by the BOS Server.

The first time it is initialized, the **flserver** creates the FLDB files in **/opt/dfslocal/var/dfs**; all database files have a root name of **fldb**. The **flserver** process also creates the **/opt/dcelocal/var/dfs/admin.fl** administrative list file if the file does not already exist.

The principals and groups in the **admin.fl** administrative list are authorized to issue commands to create server entries and fileset entries in the FLDB. The list must also include the abbreviated DFS server principals of all Fileset Database machines to allow for the proper distribution changes through the Ubik database synchronization mechanism. Because the FLDB is a replicated database, the **admin.fl** administrative list must contain the same principals and groups for all **flserver** processes in an administrative domain.

In addition, when the **flserver** is first initialized, it makes a **ubik_ServerInit** call to register its existence as a server process with the Ubik coordinator. It then listens for incoming RPCs to which to respond.

When it is started, the **flserver** creates the **/opt/dcelocal/var/dfs/adm/FlLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when the **flserver** is started, the process moves it to the **FlLog.old** file in the same directory (overwriting the current **FlLog.old** file if it exists) before creating a new version to which to append messages.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS **root** refers to a user with a **UID = 0**.

Output

If problems are encountered during initialization, the **flserver** prints error messages to the standard error output. The **flserver** process keeps an event log in **/opt/dcelocal/var/dfs/adm/FlLog**. If run with the **-verbose** option, the **flserver** process provides a detailed report on what it is doing by displaying messages on standard error.

Implementation Specifics

On OS/390 DFS, the **flserver** process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **flserver**. For further information about configuring the **flserver**, refer to “bos create” on page 485.

Related Information

Files:

admin.fl	**FlLog**

fms

Purpose

Determines tape size and end of file (EOF) mark size for a tape drive.

Usage

The **fms** command is used with the Backup System to determine the tape size and EOF mark size for the tape drive indicated with **-device**. It is primarily useful for determining information required for specifying a tape drive's parameters in the **TapeConfig** file. It can also be used to initialize a tape because it inserts file marks onto the entire tape. The Backup System then does not have to insert the file marks when it dumps information to the tape (file marks are inserted after each fileset dumped to tape). The binary file for the **fms** command resides in **dceshared/bin/fms**.

Important Note to Users

OS/390 DFS uses OS/390 services to control and dynamically allocate tape drives. Because of this, the **fms** and **TapeConfig** commands are not necessary on OS/390 DFS and not available.

fts

Purpose

Introduction to the **fts** command suite.

Command Syntax

All DFS commands have the same general structure:

command {-option1 *argument***... | -option2 {***argument1* **|** *argument2***}...} [-optional_information]**

The following example illustrates the elements of a DFS command:

fts release -fileset {*name* **|** *ID***} [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

The following list summarizes the elements of a DFS command:

- **Command** - A command consists of the command suite (**fts** in the previous example) and the command name (**release**). The command suite and the command name must be separated by a space. The command suite specifies the group of related commands to which the command belongs; it indicates which program or server process executes the command. The command name directs the server process or program to perform an action.
- **Options** - Command options always appear in bold type in the text, are always preceded by a - (dash), and are often followed by arguments. In the previous example, **-fileset** is an option, with *name* or *ID* as its argument. An option and its arguments tell the server process or program which entities to manipulate when executing the command (for example, which file, which File Server machine, or which cell). In general, the issuer should provide the options for a command in the order detailed in the documentation. The { | } (braces separated by a vertical bar) indicate that the issuer must enter either one option or the other (**-noauth** or **-localauth** in the previous example).
- **Arguments** - Arguments for options always appear in italic type in the text. The { | } indicate that the issuer must enter either one argument or the other (*name* or *ID* in the preceding example). The ... (ellipsis) indicates that the issuer can enter multiple arguments.
- **Optional** information - Some commands have optional, as well as required, options and arguments. Optional information is enclosed in [ ] (brackets). All options except **-fileset** in the previous example are optional. Note that, because they are enclosed in both [ ] (brackets) and { } (braces), either the **-noauth** or the **-localauth** option can be entered.

Options

The following options are used with many **fts** commands. They are also listed with the commands that use them.

-fileset
Specifies the fileset to use with the command. You can specify either a fileset name or a fileset ID.

-server *machine*
Specifies the File Server machine to use with the command. This option is typically used to provide the name of the file server machine on which the fileset or filesets to use with the command reside. You can use any of the following to specify the File Server machine:

- 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**).

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition to use with the command. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file.

-cell *cellname*
: Specifies that the command is to be run with respect to the cell named by the *cellname* argument. By default, commands are executed in the local cell of the issuer of the command.

-noauth Directs the **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. Generally, the **-noauth** option is included with a command if DFS authorization checking is disabled on a server machine on which administrative privilege is required or if the Security Service is unavailable. If DFS authorization checking is disabled, DFS processes require no administrative privilege to issue any command; any user, even the identity **nobody**, has sufficient privilege to perform any operation. If the Security Service is unavailable, a user's security credentials cannot be obtained.

DFS authorization checking is disabled with the **bos setauth** command or by including the **-noauth** option when the **bosserver** process is started on a machine. DFS authorization checking is typically disabled

- During initial DFS installation
- If the Security Service is unavailable
- During server encryption key emergencies
- To view the actual keys stored in a keytab file.

Include the **-noauth** option with a command that requires administrative privilege only if DFS authorization checking is disabled on the necessary machines. A command that requires administrative privilege is not successful if the **-noauth** option is included and DFS authorization checking is not disabled. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, **/.../abc.com/hosts/fs1/dfs-server**. Do not confuse a machine's DFS server principal with its unique self identity.) Use this option only if the command is issued from a DFS server machine. You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs the **fts** program to provide detailed information about its actions as it executes the command. This is useful mainly for debugging or trace purposes. The amount of additional information displayed when the **-verbose** option is specified varies for different commands.

-help Prints the online help for this command. All other valid options specified with this option are ignored. For complete details about receiving help, see “Receiving Help” on page 406.

Usage

Most **fts** commands are administrative-level commands used by system administrators to contact the Fileset Server and the Fileset Location Server (**flserver**). These commands are used to instruct the Fileset Server to create and delete filesets, as well as to move, replicate, and back up filesets. The Fileset Location Server automatically records in the Fileset Location Database (FLDB) any changes in fileset status and fileset location resulting from **fts** commands.

If the execution of an **fts** command is interrupted by a server or a process failure, subsequent execution of the command continues at the interruption point rather than at the beginning of the operation. Therefore, before executing a command, the Fileset Server and the FL Server verify that running the command has an effect. If the desired end-state already exists, the command is not executed; if the end-state does not yet exist, the command continues as necessary to achieve it.

If the issuer explicitly interrupts a fileset operation with an interrupt signal, the fileset is locked; the issuer must unlock it with the **fts unlock** command before proceeding.

DCE Local File System

The DCE Local File System is a high-performance, log-based file system. It supports the use of DCE Local File System aggregates, which are physically equivalent to standard UNIX disk partitions but also contain a specialized log of metadata about the structure and location of information they house.

DCE Local File System aggregates, in turn, support the use of DCE Local File System filesets. DCE Local File System filesets are hierarchical groupings of files managed as a single unit. They can vary in size but are almost always smaller than a disk partition. As a result, multiple DCE Local File System filesets can be stored on a single aggregate. Non-Local File System filesets occupy the entire partition on which they reside.

Because of the differences between DCE Local File System and non-Local File System filesets, the following **fts** commands function only with DCE Local File System filesets. Refer to the appropriate commands for more information about the functionality provided by the following commands:

fts addsite
fts clone
fts clonesys
fts create
fts delete
fts dump
fts lsreplicas
fts move
fts release
fts restore
fts rmsite
fts setquota
fts setrepinfo
fts statrepserver
fts update
fts zap.

Important Note to Users

On OS/390 DFS, the **fts dump** and **fts restore** commands cannot be issued against non-Local File System filesets.

Fileset Location Database Information

The Fileset Location Database (FLDB) is maintained by the Fileset Location Server (**flserver**). A master copy of the FLDB is stored on one Fileset Database machine, with copies synchronized on other Fileset Database Machines using the Ubik library of facilities. It is essential that the information in the FLDB correspond to the status of the filesets on the File Server machines. Therefore, any **fts** command that affects the fileset status also changes the corresponding FLDB entry automatically. If an **fts** operation is interrupted before completion, information in the FLDB can differ from information on a file server machine. In these cases, the **fts syncserv** and **fts syncfldb** commands must be used to align the information.

There is an entry in the FLDB for each read-write DCE Local File System and non-Local File System fileset. Each entry for a DCE Local File System fileset also records information about the read-only and backup versions of the fileset (these versions do not have their own entries). The information in an FLDB entry includes the fileset's name and fileset ID number, the ID numbers of its read-only and backup versions (if it is a DCE Local File System fileset), site definitions, site counts, and status flags. Complete details about the FLDB are included in Chapter 10, “Making Filesets and Aggregates Available” on page 147.

Fileset Header Information

A separate fileset header is stored at the site of each copy of a DCE Local File System fileset, regardless of its type (read-write, read-only, or backup). The data structure of the fileset header records which physical memory addresses of the files in the fileset on the partition on which the fileset is stored. The fileset header binds all the files into a logical unit without requiring that they be stored in contiguous memory blocks.

The information in the fileset header of a DCE Local File System fileset includes the following:

- The fileset's name
- The fileset ID number
- The type (read-only, read-write, or backup)
- The size
- The ID numbers of its parent, clone, and backup versions
- The fileset's creation date
- The date at which it was last updated.

Cautions

Specific cautionary information is included with individual commands.

Receiving Help

There are several different ways to receive help about DFS commands. The following examples summarize the syntax for the different help options available on OS/390 DFS:

$ **fts help** Displays a list of commands in a command suite.

$ **fts help** *command*
Displays the syntax for a single command.

$ **fts apropos -topic** *string*
Displays a short description of any commands that match the specified *string.*

Refer to “Receiving Help” on page 406 for complete information about the DFS help facilities.

Privilege Required

Most **fts** commands can be issued by users included in either the **admin.ft** file or the **admin.fl** file. Some commands require that the issuer be included on both lists; some commands also require that the user have certain permissions for a file or directory. Specific privilege information is listed with each command's description.

Implementation Specifics

On OS/390 DFS, the **fts dump** and **fts restore** commands cannot be issued against non-Local File System filesets.

Related Information

Commands:

fts addsite	**fts edserverentry**	**fts restore**
fts aggrinfo	**fts help**	**fts rmsite**
fts apropos	**fts lock**	**fts setprotectlevels**
fts clone	**fts lsaggr**	**fts setquota**
fts clonesys	**fts lsfldb**	**fts setrepinfo**
fts create	**fts lsft**	**fts statftserver**
fts crfldbentry	**fts lsheader**	**fts statrepserver**
fts crmount	**fts lsmount**	**fts syncfldb**
fts crserverentry	**fts lsquota**	**fts syncserv**
fts delete	**fts lsreplicas**	**fts unlock**
fts delfldbentry	**fts lsserverentry**	**fts unlockfldb**
fts delmount	**fts move**	**fts update**
fts delserverentry	**fts release**	**fts zap**
fts dump	**fts rename**	

Files:

admin.fl	**devtab**	**dfstab**
admin.ft		

fts addsite

Purpose

Adds a replication site for a read-write DCE Local File System fileset.

Format

fts addsite -fileset {*name* **|** *ID***} -server** *machine* **-aggregate** *name* **[-maxsiteage** *interval***] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the read-write source fileset for which the replication site is to be added.

-server *machine*
: Names the File Server machine on which the replica is to be stored. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate where the replica is to be stored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-maxsiteage *interval*
: Specifies the maximum amount of time that the replica to be stored at the site can be out of date (MaxSiteAge). The Replication Server attempts to keep the replica current within this amount of time. If this option is omitted, the **DefaultSiteAge** for the read-write site is used as the value for the MaxSiteAge. This option must be specified if the **DefaultSiteAge** was not defined for the read-write fileset (if the **-defaultsiteage** option was omitted when the **fts setrepinfo** was used to set the replication parameters for the read-write fileset).

  Use this option only with Scheduled Replication. The MaxSiteAge of a replication site is ignored if Release Replication is used.

-cell *cellname*
: Specifies the cell with respect to which the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts addsite** command defines a replication site for the read-write DCE Local File System fileset specified with the **-fileset** option. A replication site is a File Server machine and aggregate where a read-only replica of a read-write fileset is to be stored. The command also increments the number of fileset entries recorded as residing on the File Server machine specified with **-server** in the Fileset Location Database (FLDB) entry for the server.

A fileset's replication sites are recorded in the FLDB entry for the fileset. If this is the first replication site defined for the fileset, the status flag for the read-only version of the fileset is changed to `valid` in anticipation of the creation of a read-only fileset at the site.

Enter this command once for each replication site to be defined for a read-write fileset. Before this command is issued, the **fts setrepinfo** command must be used to set the replication parameters for the read-write fileset and a server entry must exist in the FLDB for the File Server machine specified with **-server**.

If Release Replication is used with the fileset, the first site defined with this command must be on the same File Server machine as the read-write, source fileset. If it is on the same aggregate as the source fileset, it is created as a clone of the source. Because it is created as a clone fileset, which has the same structure as a backup fileset, it shares data with the read-write fileset. Therefore, it requires potentially much less space than a full read-only fileset created on a different aggregate.

A File Server machine can house only a single read-only version of a given read-write fileset. The command fails if an attempt is made to define a second replication site for a given fileset on the same File Server machine. Also, the File Server machine that houses a replication site must reside in the same cell as the read-write fileset for which the replication site is defined. The FLDB entry for a read-write fileset records the locations of the fileset's replication sites; the server machine on which a site is defined must have a server entry in the FLDB that records the entry for the read-write fileset.

The FLDB can record a maximum of 16 sites for all versions of a fileset combined, a site being a specific File Server machine and aggregate. The read-write version and backup version (if it exists) of a fileset share a single site definition. If you define a replication site for a fileset at the same site as its read-write and backup versions, you can then define 15 more replication sites for the fileset; this approach allows you to define up to 16 replication sites. If you do not place a replica of a fileset at the same site as its read-write and backup versions, you can define a maximum of 15 replication sites for the fileset.

The **-maxsiteage** option is used to define the MaxSiteAge for the site, which is the maximum amount of time the replica at the site can be out of date. The Replication Server always attempts to copy the latest version of the fileset to the site before the MaxSiteAge expires. Use the **-maxsiteage** option only if Scheduled Replication is used with the read-write source fileset; the MaxSiteAge is ignored if Release Replication is used.

The DefaultSiteAge associated with the read-write fileset is used by default if the **-maxsiteage** option is omitted. The **-maxsiteage** option is required with the **fts addsite** command if the **-defaultsiteage** option was omitted when the **fts setrepinfo** command was used to define the replication parameters for the read-write fileset.

If Release Replication is used, the **fts release** command must be used to place a read-only replica at the replication site defined on the same File Server machine as the source fileset. The Replication Servers at the fileset's other replication sites then copy the replica to the sites on their respective machines. If Scheduled Replication is used, the Replication Servers at the replication sites automatically copy the source fileset to their sites. Immediate updates to sites using either type of replication can be forced with the **fts update** command.

Use the **fts aggrinfo** command to check the available space on an aggregate before adding it as a replication site. (Use the **fts lsft** command to check the size of the read-write fileset.) Use the **fts rmsite** command to remove a replication site and to instruct the Replication Server to remove the replica stored at the site. Use the **fts lsreplicas** command to determine the status of the read-only replica at a site.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or must own the server entry for each machine on which a version of the source fileset specified with **-fileset** resides.

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Examples

The following command defines a read-only site for the fileset **larryft**. The site is defined as the aggregate **lfs8** on the File Server machine named **dcefvt8**.

```
$ fts addsite larryft -server dcefvt8 -aggregate lfs8
```

Related Information

Commands:

fts lsreplicas	**fts rmsite**	**fts update**
fts release	**fts setrepinfo**	

File:
dfstab

fts aggrinfo

Purpose

Displays disk space information about exported aggregates and partitions exported from a File Server machine.

Format

fts aggrinfo -server *machine* **[-aggregate** *name***] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
Names the File Server machine about whose aggregates and partitions information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of an exported aggregate or partition about which information is to be displayed. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. If this option is omitted, information is provided about all of the exported aggregates and partitions on the specified machine.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts aggrinfo** command lists information about the total amount of disk space and the amount of disk space currently available on exported aggregates and partitions. The **-server** option is used to specify the File Server machine on which the aggregates and partitions reside. The **-aggregate** option can be used to specify a single aggregate or partition about which information is to be displayed. If this option is omitted, information about all aggregates and partitions exported from the specified server is displayed. (Much of the information displayed by the **fts aggrinfo** command is specified in the **/opt/dcelocal/var/dfs/dfstab** file.)

Privilege Required

No privileges are required.

Output

The **fts aggrinfo** command displays a separate line for each aggregate or partition. Each line displays the following information:

- The file system type (DCE Local File System or non-Local File System).
- 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.
- The number of kilobytes reserved for overdraw by some non-Local File System implementations. (Overdraw is disk space reserved in case the allocated quota of the partition is exceeded.) DCE Local File System aggregates do not reserve overdraw disk space.

Example

The following example displays information about the disk space available on all aggregates and partitions exported from the File Server machine named **/.../abc.com/hosts/fs1**:

```
$ fts aggrinfo dcefvt8

LFS aggregate lfs8 (/dev/lfs008): 12052 K free out of total 12960 (1432
reserved)
Non-LFS aggregate ufs210 (/dev/ufs210): 59616 K free out of total 60480
```

Related Information

Command:
fts lsaggr
File:
dfstab

fts apropos

Purpose

Shows each help entry containing a specified string.

Format

fts apropos -topic *string* [**-help**]

Options

-topic Specifies the keyword string for which to search. If it is more than a single word, surround it with double quotes ("") or other delimiters. Type all strings for **fts** commands in all lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts apropos** command displays the first line of the online help entry for any **fts** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **fts help** command.

Privilege Required

No privileges are required.

Results

The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **fts** command where the string specified by **-topic** is part of the command name or first line.

Examples

The following command lists all **fts** commands that have the word **mount** in their names or short descriptions:

```
$ fts apropos mount

crmount: make mount point
delmount: delete mount point
lsmount: list mount point
```

Related Information

Command:
fts help

fts clone

Purpose

Creates a backup version of a read-write DCE Local File System fileset.

Format

fts clone -fileset {*name* **|** *ID***} [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the read-write source fileset.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

This command creates a backup version, or clone, of the indicated read-write DCE Local File System fileset. It names the new backup version by adding a **.backup** extension to the name of its read-write source fileset. It places the backup version at the same site (File Server machine and aggregate) as the read-write version. The **fts clone** command *cannot* backup non-Local File System filesets.

If no backup version exists, the command changes the status flag for the backup version in the fileset's entry in the Fileset Location Database (FLDB) to `valid`. It also increments the number of fileset entries recorded as residing on the File Server machine in the FLDB entry for the server.

If a backup version already exists, the new clone replaces it. The status flag for the backup version remains `valid`, and the number of fileset entries recorded in the File Server machine's FLDB entry remains unchanged.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine on which the read-write copy of the fileset is stored. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine where a version of the fileset resides.

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Examples

The following command creates a backup version of the fileset **user.smith**:

```
$ fts clone user.smith
```

Related Information

Command:
fts clonesys

fts clonesys

Purpose

Creates backup versions of all indicated read-write DCE Local File System filesets.

Format

fts clonesys [-prefix *string*] **[-server** *machine*] **[-aggregate** *name*] **[-cell** *cellname*]
[{-noauth | -localauth}] [-verbose] [-help]

Options

-prefix *string*
Specifies a character string of any length. Every fileset with a name matching this string is cloned. Include field separators (such as periods) if appropriate. This option can be combined with **-server**, **-aggregate**, or both. Omit all three options to back up all filesets in the local cell.

-server *machine*
Specifies the File Server machine where the read-write source filesets are stored. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. This option can be combined with **-prefix**, **-aggregate**, or both. Omit all three options to back up all filesets in the local cell.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate on **-server** where the read-write source filesets are stored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file. This option can be combined with **-server**, **-prefix**, or both. Omit all three options to back up all filesets in the local cell. The **-server** option must be specified if this option is used.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts clonesys** command creates a backup version, or clone, of each indicated read-write DCE Local File System fileset. It names each backup version by adding a **.backup** extension to the name of its read-write source fileset. It places each backup version at the same site (File Server machine and aggregate) as its read-write version. The **fts clonesys** command *cannot* backup non-Local File System filesets.

If no backup version of a fileset exists, the command changes the status flag for the backup version in the fileset's entry in the Fileset Location Database (FLDB) to `valid`. It also increments the number of fileset entries recorded as residing on the File Server machine in the FLDB entry for the server.

If a backup version of a fileset already exists, the new clone replaces it. The status flag for the backup version remains `valid`, and the number of fileset entries recorded in the File Server machine's FLDB entry remains unchanged.

The **fts clonesys** command returns a **0** if all DCE Local File System filesets were successfully backed up, regardless of whether backups of any non-Local File System filesets were attempted. The command returns a **1** if one or more DCE Local File System filesets could not be backed up or if only backups of non-Local File System filesets were attempted.

By combining the **-prefix**, **-server** , and **-aggregate** options, you can create backup copies of different subsets of read-write filesets. To back up:

- All filesets in the local cell, specify no options
- All filesets in the local cell with a name beginning with the same character string (for example, **sys.** or **user.**), specify the string with the **-prefix** option
- All filesets on a File Server machine, specify the machine's name with the **-server** option
- Filesets on a specific aggregate on a File Server machine, specify both the **-server** and **-aggregate** options
- Filesets with a certain prefix on a specific File Server machine, specify both the **-prefix** and **-server** options
- Filesets with a certain prefix on a specific aggregate on a File Server machine, specify the **-prefix**, **-server**, and **-aggregate** options

Use the **fts clone** command to back up a single read-write DCE Local File System fileset. See also the **modify dfs,send dfskern,clonesys** OS/390 system command on page 329.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** files on all machines on which read-write versions of the filesets are stored. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of any fileset to be backed up resides.

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Output

If the **fts clonesys** command fails to backup either one or more DCE Local File System filesets or one or more non-Local File System filesets, the command displays the following output:

```
Total FLDB entries that were successfully backed up:
x (y failed); (z wrong aggr type)
```

The variable `x` indicates the number of DCE Local File System filesets that were successfully backed up. The variable `y` indicates the number of DCE Local File System filesets that could not be backed up. The variable `z` indicates the number of non-Local File System filesets that the command attempted to back up, but could not because of the command's inability to back up non-Local File System filesets.

Examples

The following example creates a backup version of each fileset in the local cell whose name begins with the prefix **user.**:

```
$ fts clonesys -prefix user.
```

Related Information

Command:
fts clone
File:
dfstab

fts create

Purpose

Creates a read-write DCE Local File System fileset and associated FLDB entry.

Format

fts create -ftname *name* **-server** *machine* **-aggregate** *name* [**-cell** *cellname*] [**{-noauth | -localauth}**] [**-verbose**] [**-help**]

Options

-ftname *name*
: Specifies a name for the read/write fileset. The name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. The name can be no longer than 102 characters. This does not include the **.readonly** or **.backup** extension, which is added automatically when a read-only or backup version of the fileset is created. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup filesets so you cannot specify a fileset name that ends with either of these extensions.

-server *machine*
: Names the File Server machine on which to create the new read-write fileset. A server entry for the machine must already exist. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate where the read-write fileset is to be stored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help	Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts create** command creates a read-write DCE Local File System fileset, names it as specified by **-ftname**, and places it at the site specified by **-server** and **-aggregate**. The **flserver** creates an entry for the fileset in the Fileset Location Database (FLDB) and allocates the fileset a unique ID number, which is recorded in both the fileset's FLDB entry and its fileset header. It also sets the status flag recorded for the read-write site in the fileset's FLDB entry to **valid** and increments the number of fileset entries recorded as residing on the specified file server machine in the FLDB entry for the server. A server entry must exist in the FLDB for the File Server machine before this command is issued.

The **flserver** also allocates and stores in the entry for the fileset in the FLDB the fileset ID numbers for the read-only and backup versions that can be created later. It does not create these types of filesets or place anything at a read-only or backup site, so the status flags for the read-only and backup versions are set to **invalid**.

If this command succeeds, the fileset is available for use. It must be mounted in the file system with the **fts crmount** command for its contents to be visible in the global namespace. The command creates an empty root directory in the fileset, which becomes visible when the fileset is mounted. It records null ACLs as the default for use by the directory. (Although, due to the interaction between ACLs and UNIX mode bits, the directory has a set of implicit initial ACLs granting permission to different users and groups).

When a cell is initially configured, the **fts create** command can be used to create the cell's main read-write fileset, **root.dfs**. Although **root.dfs** can be a non-Local File System fileset, it must be a DCE Local File System fileset if functionality such as replication is to be available in the cell. See the “Creating Read-Only DCE Local File System Filesets” on page 181 for instructions for configuring the root fileset to support replication.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine specified by **-server**. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for the machine specified by **-server**.

Examples

The following command creates the read-write fileset **user.pat**. The fileset is created on the aggregate **/dev/ufs** on the File Server machine **fs4**.

```
$ fts create user.pat /.../abc.com/hosts/fs4 /dev/ufs
```

Related Information

Commands:

fts crfldbentry	**fts crmount**

File:
dfstab

fts crfldbentry

Purpose

Creates a fileset entry in the FLDB.

Format

fts crfldbentry -ftname *name* **-server** *machine* **-aggrid** *ID* **[-cell** *cellname*] **[{-noauth | -localauth}] [-verbose] [-help]**

Options

-ftname *name*
Specifies a name for the fileset. The name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. The name can be no longer than 102 characters. (Fileset names are restricted to this limit to accommodate the **.readonly** and **.backup** extensions that DCE Local File System filesets of those types receive. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup DCE Local File System filesets, so you cannot specify a fileset name that ends with either of these extensions.)

For filesets that are HFS or RFS filesets, if you use the OS/390 data set name as the fileset name, it can make administration easier in some cases. For example, if you need to use the **fts syncfldb** command because the FLDB has no fileset entry for this fileset (refer to the chapter on deconfiguring DFS in the *OS/390 Distributed File Service DFS Configuring and Getting Started* book, for information on renaming a machine's DCE host name), the command will create a fileset entry in the FLDB with the OS/390 data set name as the fileset name.

-server *machine*
Names the File Server machine on which the fileset resides. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggrid *ID*
Specifies the aggregate ID number to be assigned to the aggregate or partition in the Fileset Location Database (FLDB). The number specified with this option must also be used as the aggregate ID in the fourth field of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file on the machine where the aggregate or partition resides. The ID number must be a positive integer. If the command is being used to create an FLDB entry for a non-Local File System fileset (its typical use), the specified number must not already be in use in the **dfstab** file on the specified **-server**.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts crfldbentry** command is used to register a fileset in the FLDB. After the fileset is registered, its location can be tracked by the **flserver**. The command is typically used to create FLDB entries for non-Local File System filesets.

Use the **-ftname** option to specify a name for the fileset according to the guidelines presented with the description of the **-ftname** option. Use the **-server** option to indicate the file server machine that houses the aggregate or partition on which the fileset resides. Use the **-aggrid** option to specify an aggregate ID number to be associated with the aggregate or partition in the FLDB. This same number must be used in the entry for the aggregate or partition in the **dfstab** file on **-server**; choose a number that is not already in use in the machine's **dfstab** file.

The FL Server allocates a unique fileset ID number for the fileset. This number, along with ID numbers allocated for read-only and backup filesets, is returned by the command. When creating an entry for a non-Local File System fileset, the ID number allocated for the read-write fileset must be specified in the fifth field of the entry in the **dfstab** file for the partition on which the fileset resides.

The FL Server also sets the status flag for the read-write version in the fileset's entry to **valid**. In addition, it increments the number of fileset entries recorded as residing on the specified File Server machine in the FLDB entry for the server. A server entry must exist in the FLDB for the File Server machine before this command is issued.

After issuing this command to register a non-Local File System fileset, create an entry for the partition on which the fileset resides in the local **dfstab** file, export the partition with the **dfsexport** command, and mount with the **fts crmount** command to make the fileset accessible in the DCE namespace. The **fts crserverentry** command must be used before this command to create a server entry in the FLDB for the machine on which the fileset resides.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for the machine specified by **-server**.

Examples

The following example creates an entry in the FLDB for a non-Local File System fileset named **user.jlw**. The fileset is located on the File Server machine named **fs3**. The aggregate ID of the partition on which the fileset resides is **7**.

```
$ fts crfldbentry user.jlw /.../abc.com/hosts/fs3 7
```

Related Information

Commands:

dfsexport	**fts crserverentry**	**fts crmount**
fts create		

File:
dfstab

fts crmount

Purpose

Creates a mount point for a fileset.

Format

fts crmount -dir *directory_name* **{-fileset {***name* **|** *ID***} [-rw] [-fast] [-help]**

Options

-dir *directory_name*
: Names the location in the file tree at which the root directory of the fileset is to be mounted (also known as the mount point). The specified name must not already exist. However, the parent directory of the mount point must exist in the DFS filespace.

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the fileset to be mounted. The mount point for the fileset is created with the **-dir** option. The read-write, read-only, or backup version of the fileset can be named.

-rw
: Specifies the type of the mount point as read-write. By default, a mount point is regular. If this option is used, the **-fileset** option must specify the read-write version of the fileset.

  An important function of the **-rw** option is to mount a cell's main read-write fileset, **root.dfs**, below the top level of the cell's DFS filespace. The option must be used in this capacity at installation if replication is to be available in the cell. See “Usage” for more information about using this command with the **-rw** option to create a read-write mount point for **root.dfs**

-fast
: Directs **fts** not to verify the existence of the fileset indicated with the **-fileset** option. Use this option to create a mount point for a fileset that does not yet exist.

  By default, **fts** contacts the local Fileset Location Server (**flserver**) for the cell housing the parent directory of the name supplied with the **-dir** option to verify that the FLDB contains an entry for the fileset to be mounted. If no FLDB entry exists for the specified fileset, the command displays a warning. The specified mount point is always created, regardless of whether the **-fast** option is provided; the option simply suppresses the check and any possible warnings.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts crmount** command creates a mount point for the fileset specified with the **-fileset** option at the location in the file tree specified with the **-dir** option. The mount point makes the contents of the specified fileset visible to other users. Once this command is used to mount a fileset, the fileset never needs to be mounted again; it is mounted automatically.

A mount point is actually a special symbolic link that acts as an association between a directory location and a fileset. Mount points look and function like standard directories. When the Cache Manager encounters a mount point in a pathname transversal, it determines which fileset is associated with the mount point. When it finds that fileset, it can access files or directories contained in the fileset's root directory. It traverses any paths leading through directories or other mount points in the fileset until it finds the directory or file indicated by the pathname.

To a large extent, the type of a mount point determines the version of a fileset through which the Cache Manager searches for a requested directory or file. By default, every mount point is a regular mount point. However, if the **-rw** option is included with the command, the new mount point is a read-write mount point.

When the Cache Manager encounters a mount point during its search for a directory or file, it determines which fileset is associated with the mount point. When it finds the root of that fileset, it traverses any paths leading through directories or other mount points in that fileset until it finds the indicated directory or file. The type of the mount point determines the version of a fileset through which the Cache Manager searches for the desired directory or file.

When the Cache Manager encounters a regular mount point, it checks the version of the fileset the mount point indicates. If the mount point indicates a read-only or backup version, the Cache Manager accesses that version. If it indicates the read-write version, the Cache Manager attempts to access a read-only fileset if the fileset in which the mount point resides is read-only.

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 attempts to access a read-only version of the fileset first. If no read-only versions exist, the Cache Manager attempts to access the read-write version of the fileset. If one or more read-only versions exist but all are unavailable (perhaps because of one or more machine outages), the Cache Manager cannot access the fileset because it does not attempt to access the read-write version.

When the Cache Manager encounters a read-write mount point, it attempts to access only the read-write version of the fileset, regardless of the type of fileset in which the mount point resides. If the read-write version of the fileset does not exist or is inaccessible, the Cache Manager cannot access the fileset.

Regular mount points promote greater fileset availability because they allow the Cache Manager to access read-only filesets as often as possible. Because multiple copies of read-only filesets typically exist, regular mount points generally increase fileset availability. Because only a single version of a read-write fileset can exist, read-write mount points generally reduce fileset availability.

You typically mount filesets with regular mount points. A regular mount point is explicitly not a "read-only" mount point. Although the Cache Manager can still attempt to access a read-only version of a fileset when it encounters a regular mount point, it accesses the read-write version of the fileset if no read-only versions exist.

During a cell's configuration, an important function of the **fts crmount** command is to create a mount point for the cell's main read-write fileset, **root.dfs** The command must be used with the **-rw** option to create an explicit read-write mount point for the fileset below the top level of the cell's DFS filespace. The mount point for the fileset must be created at **/.../***cellname***/fs/.rw**

The **root.dfs** fileset is the first fileset mounted in a cell. The cache manager automatically mounts it at the top-level of the cell's DFS filespace (**/.../***cellname***/fs** by default, but it can be defined as something else). It must be created as a DCE Local File System fileset with the **fts create** command if functionality such as replication is to be available in the cell.

Once the **root.dfs** fileset is mounted with a read-write mount point, it can be replicated. Replication is then available for DCE Local File System filesets in the cell. If **root.dfs** is replicated before its read-write mount point is created with this command, the read-write version of **root.dfs** cannot be accessed in the cell. “Creating Read-Only DCE Local File System Filesets” on page 181 provides instructions for configuring the root fileset to support replication.

Privilege Required

If the parent directory of the mount point (the directory in which the mount point is to be created) is in a DCE Local File System fileset, the issuer must have the write, execute, control, and insert permissions on the directory. If the directory is in a non-Local File System fileset, the issuer must have write and execute permissions on the directory.

Note: You cannot create a mount point in an RFS fileset.

Cautions

Do not mount a fileset at more than one location in the file system. Creating multiple mount points can distort the hierarchical nature of the file system. Because the Cache Manager stores only a single pointer to the parent directory of the mount point for each fileset, it can become confused about which pathname to follow when searching for a file in a fileset with multiple mount points. This is true even if the full pathname of a file is specified.

Create multiple mount points for a fileset sparingly (only in a very limited number of troubleshooting and testing situations). Remove the extraneous mount points as soon as they are no longer necessary.

Do not create a symbolic link that begins with a # (number sign) or a % (percent sign) character. Because a mount point is a special type of symbolic link that uses these characters internally to identify its type, the Cache Manager becomes confused if it encounters a normal symbolic link that begins with one of these characters.

Examples

The following command creates a regular mount point (the default type of mount point) for a read-write fileset named **user.jlw** at the directory named **/.../abc.com/fs/usr/jlw**:

```
$ fts crmount /.../abc.com/fs/usr/jlw user.jlw
```

The final command creates a read-write mount point for the **root.dfs** fileset in the **abc.com** cell. The fileset is mounted at **.rw**, below the top-level of the cell's DFS filespace.

```
$ fts crmount /.../abc.com/fs/.rw root.dfs -rw
```

Related Information

Commands:

dfsd	**fts delmount**	**fts lsmount**
fts create		

fts crserverentry

Purpose

Creates a server entry in the FLDB.

Format

fts crserverentry -server {*machine* **|** *address***} -principal** *name* **[-quota** *entries***] [-owner** *group***]**
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server {*machine* **|** *address***}**
: Specifies the server machine for which an entry is to be created in the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-principal *name*
: Specifies the abbreviation for the DFS server principal to be registered in the FLDB for the machine (for example, **hosts/***hostname*). The machine's principal name in the Registry Database must match this name.

-quota *entries*
: Sets a limit on the number of fileset entries (read-write, read-only, and backup) in the FLDB that can be associated with the specified **-server**. If this option is omitted, a value of 0 (zero) is used, meaning an unlimited number of fileset entries can be associated with the file server machine.

-owner *group*
: Specifies the name of the group that is the owner of the server entry. A group can be specified by a full or abbreviated group name (for example, **/.../***cellname***/***group_name* or just *group_name*). Foreign groups cannot own a local server entry. If this option is omitted, no group owns the server entry (the value <**nil**> is recorded as the owner).

-cell *cellname*
: Specifies the cell in whose FLDB the server entry is to be created. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts crserverentry** command creates a server entry in the FLDB for the server machine specified with **-server**. You must issue this command for a server machine before issuing any other **fts** commands involving that machine (for example, before creating filesets on the machine with the **fts create** command, before adding the machine as a replication site with the **fts addsite** command, before moving filesets to the machine with the **fts move** command, and so on).

The **-quota** option is used to limit the number of filesets (read-write, read-only, and backup) that can reside on the specified File Server machine. When a fileset entry in the FLDB is defined to reference the file server machine as the site of a version of a fileset, the **flserver** increments the number of fileset entries recorded as residing on the server in its server entry. The **flserver** creates no more than the specified **-quota** of fileset entries on the server machine.

The following commands update the number of fileset entries recorded for a File Server machine in its server entry: The **fts create**, **fts crfldbentry**, **fts addsite**, **fts restore**, **fts clone**, and **fts clonesys** commands increment the number of fileset entries recorded for the server; the **fts delete**, **fts delfldbentry**, and **fts rmsite** commands decrement the number of fileset entries recorded; the **fts move** command decrements and increments the number of fileset entries recorded on the source and destination machines, respectively; and the **fts syncfldb** and **fts syncserv** commands can update the number of fileset entries recorded, as necessary.

The **-owner** option is used to specify a group of administrators who can administer entries in the FLDB for all of the filesets on the specified File Server machine. The same group can be given ownership of all server entries for the file server machines in the domain where the specified machine resides. Members of the group can then manipulate the FLDB entries for all of the filesets in the domain where the File Server machine resides. This way, the administrators in the group need not be included on the **admin.fl** list for the entire cell, which would allow them to modify all of the fileset entries in the FLDB in that cell.

Use the **fts lsserverentry** command to display the current values from the FLDB for a server entry. Use the **fts edserverentry** command to change the current values in the FLDB for a server entry. Use the **fts delserverentry** command to remove a server entry from the FLDB.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines.

Examples

The following example adds a server entry to the FLDB for a server machine named **DCEFVT8**. The abbreviated DFS server principal of the machine is specified with the **-principal** option as **hosts/DCEFVT8**. Because they are omitted, the **-quota** and **-owner** options receive the default values of 0 (zero) and the empty group ID (displayed as <**nil**>), respectively. Note that the *hostname* entry for a server machine is supplied in upper case letters.

```
$ fts crserverentry dcefvt8  hosts/DCEFVT8
```

Related Information

Commands:

fts delserverentry	**fts edserverentry**	**fts lsserverentry**

fts delete

Purpose

Removes a specified read-write or backup version of a DCE Local File System fileset.

Format

fts delete -fileset {*name* **|** *ID***} -server** *machine* **-aggregate** *name* **[-cell** *cellname***]**
[{-noauth | -localauth}] [-verbose] [-help]

Options

-fileset {*name* **|** *ID***}**
Specifies the complete name or fileset ID number of the read-write or backup fileset to be removed. Include the **.backup** extension if specifying the name of a backup fileset.

-server *machine*
Names the File Server machine from which to remove the fileset. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate from which to remove the fileset. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-cell *cellname***]**
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts delete** command removes the read-write or backup DCE Local File System fileset indicated by the **-fileset** option from the site specified by the **-server** and **-aggregate** options. Versions of the fileset are removed and the Fileset Location Database (FLDB) entry for the fileset updated to record the removals as follows:

- Removing a read-write fileset automatically removes its associated backup version (if the backup version exists). If read-only versions of the fileset exist, site information for the read-write and backup versions of the fileset is removed from the FLDB entry, and the status flags for both versions are set to **invalid** (their fileset ID numbers are still recorded); the read-only versions of the fileset are not affected. If no read-only versions of the fileset exist, the entire entry for the fileset is removed from the FLDB.

- Removing a backup fileset removes site information for the backup version from the fileset's FLDB entry and marks the backup version as **invalid** but does not erase its fileset ID number. Read-write and read-only versions of the fileset are not affected.

The number of fileset entries recorded in the server entry in the FLDB for the File Server machine from which a read-write or backup version of a fileset is removed is decremented once for each deleted version of the fileset. (Note that, if the indicated version of a fileset does not exist at the specified site but is referenced in the fileset's FLDB entry, the command removes site information about that version of the fileset from the fileset's entry and generally performs all other operations as indicated.)

Before you remove the read-write (and backup) version of a fileset, use the **fts rmsite** command to remove the fileset's replication sites and to instruct the replication server to remove the replicas stored at the sites. If Release Replication was used for the fileset, use the **fts rmsite** command to remove the replication site and replica stored at the read-write fileset's site as well. After removing a fileset, use the **fts delmount** command to remove its mount point. Note that it might be better in some cases to remove a fileset's mount point before deleting the fileset; removing the mount point first ensures that no users are accessing the fileset when it is deleted.

If the DCE Local File System fileset to be removed is also mounted locally (as a file system on its File Server machine), you must remove its local mount point before you delete it; the **fts delete** command cannot be used to delete a fileset that is mounted locally. In addition, because the backup version of a fileset is removed when its read-write version is removed, you cannot remove the read-write version of a fileset if its backup version is mounted locally. You must remove the backup version's local mount point before deleting the read-write version.

Note: You cannot locally mount a DCE Local File System fileset on OS/390 DFS.

The **fts delfldbentry** command can be used to remove an FLDB entry for a fileset. Use the command only when you are certain that a fileset deletion was not recorded in the FLDB. The **fts zap** command can be used to remove a fileset when it is urgent that the fileset be removed but the FLDB is inaccessible. When the FLDB is again accessible, use the **fts delfldbentry** command to remove the fileset's FLDB entry or use the **fts syncfldb** and **fts syncserv** commands to synchronize the FLDB with the state of the filesets.

The **fts delfldbentry** command is also used to remove the FLDB entry for a non-Local File System fileset. The **fts delmount** command is used to remove its mount point, and the **dfsexport** command is used to detach the partition where it resides from the global namespace.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine specified by **-server**. The issuer must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine where a version of the fileset to be deleted resides.

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Output

The **fts delete** command displays the following error message if you attempt to delete a read/write fileset for which one or more replication sites exist:

```
There is a RO copy in existence - please 'rmsite' before 'delete'.
```

Use the **fts rmsite** command to remove all replication sites for the fileset, and reissue the **fts delete** command.

Examples

The following command deletes the read-write fileset named **user.terry** and its backup version (if it exists) from the aggregate named **/dev/ufs1** on the File Server machine named **fs4**:

```
$ fts delete user.terry /.../abc.com/hosts/fs4 /dev/ufs1
```

Implementation Specifics

You cannot locally mount a DCE Local File System fileset on OS/390 DFS.

Related Information

Commands:

dfsexport	**fts rmsite**	**fts syncserv**
fts delfldbentry	**fts syncfldb**	**fts zap**
fts delmount		

File:
dfstab

fts delfldbentry

Purpose

Removes a specified entry from the FLDB.

Format

fts delfldbentry {-fileset {*name* **|** *ID***} | -prefix** *string***} [-server** *machine***] [-aggregate** *name***]**
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of a fileset. The entire entry for the fileset is removed from the Fileset Location Database (FLDB), regardless of whether the read-write, read-only, or backup version of the fileset is specified. Provide this option or use the **-prefix** option.

-prefix *string*
: Specifies a character string of any length. Every FLDB entry that lists a fileset whose name begins with this exact string is removed (unless more restrictive constraints are specified with the **-server** and optionally **-aggregate** options). Include field separators such as periods if appropriate. Provide this option (optionally with **-server** and **-aggregate**) or use the **-fileset** option.

-server *machine*
: Names a File Server machine. If a fileset's name matches the specified **-prefix** and it is listed as residing on the specified File Server machine, its entry is removed from the FLDB. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. If this option is used, **-prefix** must be used; **-aggregate** can also be used.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of an aggregate or partition on **-server**. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. If a fileset's name matches the specified **-prefix** and it resides on the specified aggregate on **-server**, its entry is removed from the FLDB. If this option is used, **-server** and **-prefix** must be used.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts delfldbentry** command removes the entries for all indicated filesets from the FLDB. Regardless of the version of a fileset (read-write, read-only, or backup) specified with the command, the fileset's entire entry is removed. The command has no effect on actual filesets on File Server machines, only on their FLDB entries.

The command also decrements the number of fileset entries recorded in server entries, as appropriate. For each version of a fileset whose entry is removed from the FLDB, the number of fileset entries recorded in the server entry for the File Server machine on which it resides is reduced by one.

By using the **-fileset** option alone or combining the **-prefix**, **-server**, and **-aggregate** options in increasingly specific ways, FLDB entries can be removed for varying numbers of filesets. To remove the FLDB entry for:

- A single fileset, specify **-fileset**.
- Every fileset whose name begins with a certain character string (for example, **sys.** or **user.**), regardless of site, specify **-prefix**.
- Every fileset whose name begins with a certain character string and that is stored on a specific File Server machine, specify **-server** and **-prefix**.
- Every fileset whose name begins with a certain character string and that is stored on a specific aggregate of a specific File Server machine, specify **-prefix**, **-server**, and **-aggregate**.

This command can be used if the issuer is certain that a fileset removal is not recorded in the FLDB and does not want to take the time to synchronize an entire File Server machine. It can also be used to remove the FLDB entry for a non-Local File System fileset to be removed from the global namespace. (Use the **fts rmsite** command to remove an incorrect entry for a read-only site from the FLDB.)

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset whose FLDB entry is to be removed.

Cautions

Do not use this command as the standard way to remove a fileset entry from the FLDB. It can make the FLDB inconsistent with the filesets on server machines. Use the **fts delete** command to remove the fileset entry from the FLDB at the same time that the fileset is deleted. Also, because it can be used to remove multiple FLDB entries simultaneously; use this command carefully.

Examples

The following command removes the FLDB entry for the fileset **user.temp**:

```
$ fts delfldbentry user.temp
```

The following command removes all FLDB entries for filesets whose names begin with **test** and that are stored on the File Server machine named **fs3**

```
$ fts delfldbentry -prefix test -server /.../abc.com/hosts/fs3
```

Related Information

Commands:

fts clone	**fts rmsite**	**fts syncserv**
fts delete	**fts syncfldb**	**fts zap**

File:
dfstab

fts delmount

Purpose

Removes a mount point.

Format

fts delmount -dir *directory_name*... **[-help]**

Options

-dir *directory_name*
: Names the mount point to be deleted. Provide a complete pathname. The last element in the pathname must be an actual name, not . (dot) or .. (dot dot).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts delmount** command removes the mount point specified by **-dir**. The associated fileset is not affected, but it is inaccessible if no other mount points exist for it. When the mount point for a fileset is removed, any fileset mounted only as a subdirectory of the fileset's root directory becomes inaccessible.

Privilege Required

If **-dir** resides in a directory in a DCE Local File System fileset, the issuer must have write, execute, and delete permissions on the directory in which it resides. If **-dir** resides in a directory in a non-Local File System fileset, the issuer must have write and execute permissions on the directory in which it resides.

Examples

The following command removes the mount point for the fileset **user.vijay**, which is mounted at **/.../abc.com/fs/usr/vijay**:

```
$ fts delm /.../abc.com/fs/usr/vijay
```

Related Information

Commands:

fts crmount	**fts lsmount**

fts delserverentry

Purpose

Deletes a server entry from the FLDB.

Format

fts delserverentry -server *machine* **[-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Specifies the server machine whose entry is to be removed from the Fileset Location Database (FLDB). Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-cell *cellname*
: Specifies the cell from whose FLDB the server entry is to be removed. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts delserverentry** command removes the existing server entry from the FLDB for the server machine specified with **-server**. When the command is issued, no fileset entries in the FLDB can reference the server entry to be removed as the site of a fileset. If a fileset entry in the FLDB references the server entry to be removed, the command fails.

Use the **fts crserverentry** command to create a server entry in the FLDB. Use the **fts lsserverentry** command to display the current values from the FLDB for a server entry. Use the **fts edserverentry** command to change the current values in the FLDB for a server entry.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines.

Examples

The following example deletes the server entry from the FLDB for the server machine named **dcefvt8**. No filesets can reside on the machine when the command is issued. Note that the *hostname* entry for a server machine must be supplied in upper case letters.

```
$ fts delserverentry dcefvt8
```

Related Information

Commands:

fts crserverentry	**fts edserverentry**	**fts lsserverentry**

fts dump

Purpose

Converts a fileset to a byte stream format and places it in a file.

Format

fts dump -fileset {*name* **|** *ID***} {-time {***date* **|** *0***} | -version** *number***} [-server** *machine***] [-file** *filename***] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset {*name* **|** *ID***}**
Specifies the complete name or fileset ID number of the fileset to be dumped. The read-write, read-only, or backup version of the fileset can be dumped. Append the **.readonly** or **.backup** extension to the name of the fileset to dump the read-only or backup version instead of the read-write version.

-time *date* **|** *0*
Specifies a full or incremental dump. There are three legal values:

0 (zero) A **0** (zero) value causes a full dump of the current version of the fileset.

*mm***/***dd***/***yy* A month/day/year value causes an incremental dump from 12:00 a.m. on the indicated date; for example, **1/23/90** or **10/7/89**. Only files with modification time stamps equal to or later than the specified date and time are dumped.

"*mm***/***dd***/***yy hh***:***mm***" | "***mm***/***dd***/***yyyy hh***:***mm***"**
An exact time and date value causes an incremental dump from the specified time on the indicated date. Only files with modification time stamps equal to or later than the specified date and time are dumped. The time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The date format is the same as for a date alone. Surround the entire argument with "" (double quotes) because it contains a space; for example, **"1/23/90 22:30"** or **"10/7/89 3:45"**.

Valid values for *yy* are 00 to 37, which are interpreted as the years 2000-2037, and 70 to 99, which are interpreted as 1970-1999. Values between 38 and 69 cannot be interpreted because the years to which they correspond (2038-2069) exceed the capacity of the standard UNIX representation of dates (the number of seconds since 12:00 a.m. on 1 January 1970). Values between 38 and 69 are reduced to 2038. For a *yyyy* specification, valid values for *yyyy* are 1970-2069. However, values between 2038-2069 are reduced to 2038.

If specified, the time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). The default time is 00:00 (12:00 a.m.).

If a value is entered for the year (*yy*) that falls between the range of 38 and 69, a random time is generated for the default time.

Use this option to perform a full dump or to perform an incremental dump of only those files in the fileset modified since a specific date or date and time. Use this option or use **-version**.

-version *number*
Specifies an incremental dump based on the indicated fileset version number. Each DCE Local File System fileset has a version number. Each file in the fileset records the version number that was current when the file was last modified. If this option is specified, only those files with version numbers equal to or greater than the specified version number are dumped.

(A DCE Local File System fileset's version number is recorded in its fileset header; it has the same format as a fileset ID number.

Use the **fts lsheader** or **fts lsft** command to display a fileset's current version number. Use this option or use **-time**. Use this option only with DCE Local File System filesets.

-server *machine*
Names the file server machine that houses the version of the fileset to be dumped. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

This option is useful for dumping a particular read-only replica of a DCE Local File System fileset for which multiple replicas exist. If you include the **.readonly** extension with the name of a fileset specified with the **-fileset** option, or if you specify the ID number of the read-only version of a fileset with the **-fileset** option, you can use the **-server** option to indicate the machine that houses the specific replica to be dumped. If you omit the **-server** option in these cases, the command dumps the replica that resides at the fileset's oldest read-only site (the replica at the site that has been defined for the longest time).

This option is always unnecessary if the read-write or backup version of a fileset is to be dumped.

-file *filename*
Specifies the complete pathname of the file where the dump is to be written. If a complete pathname is not specified, the file is written to the current working directory. If this option is omitted, the data is sent to standard output (**STDOUT**).

Note: The **-file** option on OS/390 DFS refers to an OS/390 HFS file.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts dump** command converts the contents of the indicated fileset to a byte stream format. It puts the converted contents into the file specified with the **-file** option; if this option is omitted, the dumped data is sent to **STDOUT**. Both non-Local File System and read-write, read-only, and backup DCE Local File System filesets can be dumped.

Note: On OS/390 DFS, you cannot dump from a non-Local File System fileset that resides on an OS/390 system.

The options for this command can be used to perform the following types of dumps:

- A value of 0 (zero) specified with the **-time** option causes a full dump of the fileset.

- A date specified with the **-time** option causes an incremental dump of all files modified since 12:00 a.m. on that date.
- A date and time specified with the **-time** option cause an incremental dump of all files modified since that date and time.
- A version number specified with **-version** causes an incremental dump of all files in the fileset with version numbers equal to or greater than that version of the fileset.

Dumping a fileset does not affect its status in the Fileset Location Database (FLDB) or at the site from which it is dumped. However, it does make a fileset inaccessible for the duration of the dump operation. For this reason, it is customary to dump the backup version of a fileset to prevent the read-write version from being inaccessible for an extended time.

If a read-only replica of a DCE Local File System fileset is to be dumped and multiple replicas of the fileset exist, the **-server** can be used to name the file server machine that houses the specific replica to be dumped. Indicating a specific replica can be useful if network or hardware problems caused only some of a fileset's replicas to be updated. It can be especially useful for restoring the read-write version of a fileset that was lost before all of its replicas were updated. You can dump and restore a specific replica that was updated before the read-write version was lost. (By default, all replicas of the same fileset are always identical; to determine whether all replicas of a fileset are the same, use the **fts lsft** command to display information about specific replicas.)

The **fts restore** command can be used to restore a fileset dumped with the **fts dump** command. You can use the **fts restore** command to restore a dump file to any type of fileset (DCE Local File System or non-Local File System), regardless of the type of fileset from which it was created. Thus, you can dump and restore data between DCE Local File System and non-Local File System filesets, as well as between different types of non-Local File System filesets. See the **fts restore** command for more information about dumping and restoring filesets between different types of file systems. You cannot restore a fileset dumped in one cell to a site in another cell.

Note: On OS/390 DFS, you cannot restore to a non-Local File System fileset that resides on an OS/390 system.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine on which the fileset is stored. In addition, the issuer must have the write, execute, and insert permissions on the directory in which the dump file is to reside.

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Examples

The following command executes a full dump of the fileset **user.terry** into the file named **/tmp/terry.dump**:

```
$ fts dump user.terry -time 0 /tmp/terry.dump
```

The following command executes an incremental dump of the fileset **user.smith** into the file named **/tmp/smith.013196.dump**. Only those files in the fileset with modification time stamps equal to or later than 6:00 p.m. on 31 January 1996 are included in the dump.

```
$ fts dump user.smith -time "1/31/96 18:00" /tmp/smith.013196.dump
```

Implementation Specifics

The **-file** option on OS/390 DFS refers to an OS/390 HFS file.

On OS/390 DFS, exported non-Local File System filesets cannot be dumped. In addition, you cannot restore to a non-Local File System fileset on OS/390 DFS.

Related Information

Commands:

fts lsft	**fts restore**

fts edserverentry

Purpose

Edits a server entry in the FLDB.

Format

fts edserverentry -server {*machine***} [{-rmaddr | -addaddr** *address* **| -changeaddr** *address***}] [-principal** *name***] [-quota** *entries***] [{-owner** *group* **| -noowner}] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Specifies the server machine whose entry in the Fileset Location Database (FLDB) is to be edited. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. If the **-rmaddr**, **-addaddr**, or **-changeaddr** option is used with the command, specify the network address.

-rmaddr
: Removes the address specified with **-server** from the server entry identified by **-server** in the FLDB. If the name of the machine rather than one of its addresses is specified with **-server**, the command can choose one of the machine's addresses at random to be removed from the FLDB. Because this can have unpredictable results, always specify an address with **-server** when using the **-rmaddr** option. In addition, the command fails if the address to be removed is the only address present for the machine in the FLDB.

  If this option is specified, do not specify the **-addaddr** or **-changeaddr** option.

-addaddr *address*
: Adds the additional address specified with this option to the server entry specified by **-server** in the FLDB. A machine can have from one to four addresses associated with it in the FLDB. The command fails if you attempt to add a fifth address for the machine to the FLDB.

  If the name of the machine rather than one of its addresses is specified with **-server**, the command can choose one of the machine's addresses in the FLDB at random to have the address added to it. Because this can have unpredictable results, always specify an address with **-server** when using the **-addaddr** option.

  If this option is specified, do not specify the **-rmaddr** or **-changeaddr** option.

-changeaddr *address*
: Substitutes the address specified with this option for the address specified by **-server** in the FLDB. If the name of the machine rather than one of its addresses is specified with **-server**, the command can choose one of the machine's addresses at random to be replaced with the address specified with this option. Because this can produce unpredictable results, always specify an address with **-server** when using the **-changeaddr** option.

  If this option is specified, do not specify the **-rmaddr** or **-addaddr** option.

-principal *name*
: Changes the abbreviation for the DFS server principal that is registered for the machine in the FLDB (for example, **hosts/***hostname*). The machine's principal name in the Registry Database must match this name. If this option is omitted, the DFS server principal currently associated with the server entry remains unchanged.

-quota *entries*
: Changes the limit on the number of fileset entries (read-write, read-only, and backup) in the FLDB that can be associated with the specified **-server**. A value of 0 (zero) allows an unlimited number of entries to be associated with the server. If this option is omitted, the number of fileset entries currently allowed for the specified file server machine remains unchanged.

-owner *group*
: Changes the group that is the owner of the server entry. In the entry, the specified group replaces the current owning group, if any. A group can be specified by a full or abbreviated group name (for example, **/.../***cellname***/***group_name* or just *group_name*). Foreign groups cannot own a local server entry. If this option is omitted, no group owns the server entry (the value <**nil**> is reported as the owner). Use this option or use the **-noowner** option; omit both options to leave the current owning group unchanged.

-noowner
: Specifies that no group owns the entry. In the entry, the empty group ID, displayed as (blank), replaces the group that currently owns the server entry; the entry is unchanged in this regard if no group presently owns the server entry. Use this option or use the **-owner** option; omit both options to leave the current owning group unchanged.

-cell *cellname*
: Specifies the cell in whose FLDB the server entry is to be modified. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts edserverentry** command alters a server entry in the FLDB for the server machine specified with the **-server** option. Use the **-rmaddr** option to remove an address associated with a server from the FLDB. Use the **-addaddr** option to add a new address for a server to the FLDB or use the **-changeaddr** option to change an address for a server in the FLDB.

The **-principal** option can be used to change the DFS server principal associated with the server entry. The **-quota** option can be used to alter the number of entries that can be associated with the file server machine in the FLDB and the **-owner** option can be used to assign a new group as the owner of the server entry (or the **-noowner** option can be used to indicate that no group owns the server entry).

Unless a value associated with a server entry is explicitly modified with this command, its current value in the FLDB remains unchanged. The values associated with a server entry are initially specified when the server entry is created with the **fts crserverentry** command. The values can then be modified at any time with the **fts edserverentry** command. Use the **fts lsserverentry** command to display the current values from the FLDB for a server entry. Use the **fts delserverentry** command to remove a server entry from the FLDB.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines.

Examples

The following command modifies the server entry in the FLDB for a server machine. The command changes the machine's network address from **191.54.206.36**, as specified with the **-server** option, to **191.54.206.46** as indicated with the **-changeaddr** option. The command also allows the server to have an unlimited number of fileset entries by providing a value of 0 (zero) with the **-quota** option.

```
$ fts edserverentry 191.54.206.36 -changeaddr 191.54.206.46 -quota 0
```

Related Information

Commands:

fts crserverentry	**fts delserverentry**	**fts lsserverentry**

fts help

Purpose

Shows syntax of specified **fts** commands or lists functional descriptions of all **fts** commands.

Format

fts help [**-topic** *string*...] [**-help**]

Options

-topic *string*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **lsft**, not **fts lsft**). If this option is omitted, the output provides a short description of all **fts** commands.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts help** command displays the first line (name and short description) of the online help entry for every **fts** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **fts apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

Output

The online help entry for each **fts** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Examples

The following command displays the online help entry for the **fts delmount** command:

```
$fts help delmount

fts delmount: delete mount point.
Usage: fts delmount -dir <directory_name>...[-help]
```

Related Information

Command:
fts apropos

fts lock

Purpose

Locks a fileset entry in the FLDB.

Format

fts lock -fileset {*name* **|** *ID***} [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the fileset whose entry in the FLDB is to be locked. All versions of the fileset referenced in the entry are affected by the lock, regardless of whether the read-write, read-only, or backup version of the fileset is specified.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lock** command locks the entry in the FLDB for the fileset indicated with the **-fileset** option. Locking a fileset's FLDB entry blocks operations on all versions of the fileset, regardless of whether the read-write, read-only, or backup version of the fileset is indicated with the **-fileset** option. Locking a fileset's entry prevents all versions of the fileset from being modified with **fts** commands.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be locked resides.

Cautions

Do not use this command in normal circumstances. It is useful only if the system administrator wants to guarantee that no one else manipulates the fileset until the lock is released and if there is reason to believe that locking will not happen automatically. Locking a fileset only inhibits operations such as deleting and cloning of the fileset; it does not prevent the reading of data from the fileset.

Examples

The following command locks the FLDB entry for **user.terry**:

```
$ fts lock user.terry
```

Related Information

Commands:

fts unlock	**fts unlockfldb**

fts lsaggr

Purpose

Lists all exported aggregates and partitions on a File Server machine.

Format

fts lsaggr -server *machine* **[-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Names the file server machine whose exported aggregates and partitions are to be listed. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsaggr** command displays information about all exported aggregates and partitions on the file server machine specified by the **-server** option. The information about each aggregate and partition is specified in the **/opt/dcelocal/var/dfs/dfstab** file on the machine.

You can also issue the **dfsexport** command with no options to list all aggregates and partitions currently exported from the local disk to the DCE namespace. You can use the **fts aggrinfo** command to display information about the amount of disk space available on a specific aggregate or partition or on all aggregates and partitions on a file server machine.

Privilege Required

No privileges are required.

Output

This command displays a separate line for each aggregate or partition. Each line displays the following information:

- The aggregate name, specified in the second field of the **dfstab** file.
- The device name, specified in the first field of the **dfstab** file.
- The aggregate ID, specified in the fourth field of the **dfstab** file.
- The file system type, specified in the third field of the **dfstab** file.

Examples

The following example shows that two non-Local File System partitions and two DCE Local File System aggregates are exported from the file server machine named **/.../abc.com/hosts/fs1**:

```
$ fts lsaggr /.../abc.com/hosts/fs1

There are 4 aggregates on the server /.../abc.com/hosts/fs1 (fs1.abc.com):
/usr (/dev/ufs2): id=3     (non-LFS)
/tmp (/dev/ufs3): id=4     (non-LFS)
lfs1 (/dev/lfs1): id=10     (LFS)
lfs2 (/dev/lfs2): id=11     (LFS)
```

Related Information

Commands:
dfsexport **fts aggrinfo**
File:
dfstab

fts lsfldb

Purpose

Shows information from fileset entries in the FLDB.

Format

fts lsfldb [-fileset {*name* **|** *ID***}] | [-server** *machine***] [-aggregate** *name***][-locked]**
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of a fileset about which information from the Fileset Location Database (FLDB) is to be displayed. Use this option or use -**server** (and optionally **-aggregate**), **-locked**, or both. Omit this option and the **-server**, **-aggregate**, and **-locked** options to display information about all fileset entries in the FLDB.

-server *machine*
: Names a File Server machine about whose filesets information from the FLDB is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. This option can be combined with **-aggregate** to display information about the filesets on a single aggregate on **-server**, or it can be combined with **-locked** to display information about the filesets with locked FLDB entries on the server machine. Use this option alone or with **-aggregate**, **-locked**, or both, or use **-fileset**. Omit this option and the **-fileset**, **-aggregate**, and **-locked** options to display information about all fileset entries in the FLDB.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** about whose filesets information from the FLDB is to be displayed. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. The **-server** option must be provided with this option. The **-locked** option can be supplied with this option to display information about the filesets with locked FLDB entries on the aggregate.

-locked
: Specifies that the output show information only for filesets with locked FLDB entries. Use this option alone to see information for all filesets with locked FLDB entries. Use this option with **-server** (and optionally **-aggregate**) to see all filesets on a specific server machine (and optionally aggregate) with locked FLDB entries. Use this option alone or with **-server** (and optionally **-aggregate**) or use **-fileset**. Omit this option and the **-fileset**, **-server**, and **-aggregate** options to display information about all fileset entries in the FLDB.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged in to the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsfldb** command formats and displays information about fileset entries from the FLDB. Its options can be combined to display information about a variety of different filesets. To display FLDB information for

- Every fileset entry, specify no options.
- Every fileset entry that mentions a specific File Server machine as the site of any version of a fileset, specify the name of the machine with **-server**.
- Every fileset entry that mentions a specific aggregate on a specific File Server machine as the site of any version of a fileset, specify both **-server** and **-aggregate**.
- The FLDB entries for filesets with locked entries, specify the **-locked** option alone or with **-server** (and optionally **-aggregate**).
- The fileset entry for a single fileset, specify the fileset name or ID number with **-fileset**.

Use the **fts lsheader** command to display information from fileset headers. To display more information about a single fileset, use the **fts lsft** command to display all of the information displayed by the **fts lsheader** command when the **-long** option is used and all of the information displayed by this command.

Privilege Required

No privileges are required.

Output

The **fts lsfldb** command displays the following information from the FLDB for each DCE Local File System fileset specified with **-fileset** or **-server** (and optionally **-aggregate**). Because functionality such as replication is not supported for non-Local File System filesets, this command displays less information for non-Local File System filesets.

- The fileset's name.
- The fileset IDs of the read-write, read-only, and backup versions of the fileset.
- For each version, a status flag of `valid` indicates the version actually exists at a site; a status flag of `invalid` indicates the version does not exist at any site. (For the read-only version, the status flag indicates whether a replication site is defined.)
- The maximum and minimum advisory RPC authentication bounds for use in communications with Cache Managers. There are two sets of bounds: one set governs communications with Cache Managers in the local cell, while the other set governs communications with Cache Managers in foreign cells. Currently, these bounds are not enforced but serve to bias the Cache Manager's initial authentication level.
- The number of sites a version of the fileset exists at.
- An indicator if the FLDB entry is locked (the indicator is omitted if the entry is not locked).
- The replication parameters associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) where read-write (RW), read-only (RO), or backup (BK) versions of the fileset reside.

- For a read-only version, the MaxSiteAge replication parameter defined for that site; for a read-write version, `0:00:00`.
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine (or **<nil>** if no group owns the server entry).

If the output includes more than one FLDB entry, information about the filesets is displayed in alphabetical order by fileset name. The last line of the output displays the total number of entries successfully reported and the total number of entries not reported (the number of entries that `failed`).

Examples

The following command shows an example of the output from the **fts lsfldb** command for a fileset named **user.terry**:

```
$ fts lsfldb user.terry
       readWrite ID  0,,196953  valid
       readOnly  ID  0,,196594  valid
       backup    ID  0,,196595  not valid
  Minimum local protection: rpc_c_protect_level_none
  Maximum local protection: rpc_c_protect_level_pkt_privacy
  Minimum remote protection: rpc_c_protect_level_none
  Maximum remote protection: rpc_c_protect_level_pkt_privacy
  number of sites: 1
    Sched repl: maxAge=2:00:00; failAge=1d0:00:00;
  reclaimWait=18:00:00; minRepDelay=0:05:00; defaultSiteAge=0:30:00
     server        flags    aggr  siteAge principal  owner
  fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Related Information

Commands:

fts lock	**fts lsft**	**fts unlockfldb**
fts lsfldb	**fts unlock**	

File:
dfstab

fts lsft

Purpose

Lists fileset information from both the fileset header and the FLDB entry.

Format

fts lsft [{-path {*filename* **|** *directory_name***} | -fileset {***name* **|** *ID***}}] [-server** *machine***]**
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-path *filename* **|** *directory_name*
: Names a file or directory on the fileset whose fileset header and FLDB information is to be displayed. Use this option or use **-fileset**; omit both options to display information about the fileset that contains the current working directory.

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the fileset to be examined. The read-write, read-only, or backup version of the fileset can be specified. Append the **.backup** or **.readonly** extension to the name of the fileset to list information about the backup or read-only version instead of the read-write version; if the read-write version no longer exists, the command fails if the **.backup** or **.readonly** extension is not used with the name of the fileset.

  Use this option or use **-path**; omit both options to display information about the fileset that contains the current working directory.

-server *machine*
: Names the File Server machine that houses the fileset about which information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

  This option is useful for displaying information about a particular read-only replica of a DCE Local File System fileset for which multiple replicas exist. If you include the **.readonly** extension with the name of a fileset specified with the **-fileset** option, specify the ID number of the read-only version of a fileset with the **-fileset** option, or specify the path to a file or directory in a read-only fileset with the **-path** option, you can use the **-server** option to indicate the machine that houses the specific replica about which information is to be displayed. If you omit the **-server** option in these cases, the command displays information about the replica at the fileset's oldest read-only site (the replica at the site that has been defined for the longest time).

  This option is always unnecessary if information is to be displayed about the read-write or backup version of a fileset.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsft** command displays information from both the fileset header and the Fileset Location Database (FLDB) entry for the specified fileset. It displays the same output as the **fts lsheader** command with the **-long** option and the **fts lsfldb** command for a single fileset. It can be used to learn the fileset ID number of a fileset or to examine locked FLDB entries.

The fileset whose information is to be displayed can be specified by indicating the name of a file or directory on the fileset with the **-path** option, or it can be specified by indicating its name or ID number with the **-fileset** option. Omit both the **-path** and **-fileset** options to display information about the fileset that contains the current working directory. If the name of the fileset is specified with the **-fileset** option, the **.backup** or **.readonly** extension can be appended to the name to display information about one of those fileset versions rather than the read-write version.

If information about a read-only replica of a DCE Local File System fileset is to be displayed and multiple replicas of the fileset exist, the **-server** option can be used to name the File Server machine that houses the specific replica about which information is to be displayed. Indicating a specific replica can be useful if network or hardware problems caused only some of a fileset's replicas to be updated. (By default, all replicas of the same fileset should always contain the same information.)

Use the **fts lsheader** command to display information from fileset headers. Use the **fts lsfldb** command to display information from fileset entries in the FLDB.

Privilege Required

No privileges are required.

Output

The **fts lsft** command displays the following information from the fileset header and the FLDB entry for a specified DCE Local File System fileset. The following information is displayed for DCE Local File System filesets. Because non-Local File System filesets do not have DCE Local File System fileset headers, and because functionality such as replication is not supported for non-Local File System filesets, this command displays less information for non-Local File System filesets.

The command displays the following information from the fileset's header:

- The fileset's name (with a **.readonly** or **.backup** extension, if appropriate).
- Its fileset ID number.
- Its type (RW for read-write, RO for read-only, or BK for backup).
- Its type (Local File System or non-Local File System).
- Information about the state of the fileset.
- Its status (`On-line`, `Off-line`, or an error indicator).
- The File Server machine, aggregate name, and aggregate ID number on which it resides.
- The ID numbers of the parent, clone, and backup filesets related to the fileset.
- The ID numbers of the low-level backing and low-level forward filesets related to the fileset.

- Its version number.
- Its allocation and allocation usage, in kilobytes.
- Its quota and quota usage, in kilobytes.
- The day, date, and time when the fileset was created (replicated or backed up for a read-only or backup fileset).
- The day, date, and time when the contents of the fileset were last updated (same as the creation time for a read-only or backup fileset).

It then displays the following information from the fileset's entry in the FLDB:

- The fileset's name.
- The fileset IDs of the read-write, read-only, and backup versions of the fileset.
- For each version, a status flag of `valid` indicates the version actually exists at a site; a status flag of `invalid` indicates the version does not exist at any site. (For the read-only version, the status flag indicates whether a replication site is defined.)
- The maximum and minimum advisory RPC authentication bounds for use in communications with Cache Managers. There are two sets of bounds: one set governs communications with Cache Managers in the local cell, while the other set governs communications with Cache Managers in foreign cells. Currently, these bounds are not enforced but serve to bias the Cache Manager's initial authentication level.
- The number of sites at which a version of the fileset exists.
- An indicator if the FLDB entry is locked. (The indicator is omitted if the entry is not locked.)
- The replication parameters associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) on which read-write (RW), read-only (RO), or backup (BK) versions of the fileset reside.
- For a read-only version, the MaxSiteAge replication parameter defined for that site; for a read-write version, `0:00:00`.
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine (or `<nil>` if no group owns the server entry).

Examples

The following example displays information from the fileset header and FLDB entry for a DCE Local File System fileset named **user.terry**:

```
$ fts lsft -fileset user.terry
```

```
user.terry 0,,196953 RW LFS     states 0x10010005 On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    429496729 K alloc limit;      1252 K alloc usage
        15000 K quota limit;      9340 K quota usage
    Creation Fri Oct 15 16:45:16 1993
    Last Update Mon Nov 22 11:36:00 1993

user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
Minimum local protection: rpc_c_protect_level_none
Maximum local protection: rpc_c_protect_level_pkt_privacy
Minimum remote protection: rpc_c_protect_level_none
Maximum remote protection: rpc_c_protect_level_pkt_privacy
number of sites: 2
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00;
  reclaimWait=18:00:00; minRepDelay=0:05:00;
defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

Related Information

Commands:

fts lsfldb	**fts lsheader**

fts lsheader

Purpose

Shows information from fileset headers.

Format

fts lsheader -server *machine* **[-aggregate** *name***] [{-fast | -long}]**
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Names a File Server machine about whose filesets header information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. This option can be combined with the **-aggregate** option to name a specific aggregate on **-server**.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** from whose filesets header information is to be displayed. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. The **-server** option must be provided with this option.

-fast Directs the output to display only the fileset ID numbers of all filesets on the indicated server (and optionally the aggregate). If you use this option, do not use the **-long** option.

-long Directs the output to display more detailed information about all filesets on the indicated server (and optionally the aggregate). If you use this option, do not use the **-fast** option.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsheader** command formats and displays information from the fileset headers of filesets on the specified server (and optionally the aggregate or partition). To display information from the headers of all filesets on a specific File Server machine, specify the name of the server machine with the **-server** option. To specify information from the headers of all filesets on a specific aggregate on a File Server machine, specify the name of the server machine with the **-server** option and the name of the aggregate or partition with the **-aggregate** option.

Include the **-fast** option with the command to display only the ID numbers of the filesets at the specified location. Include the **-long** option with the command to display more detailed information from the headers of the filesets at the specified location.

Use the **fts lsfldb** command to display information from fileset entries in the Fileset Location Database (FLDB). To display more information about a single fileset, use the **fts lsft** command to display all of the information displayed by this command when the **-long** option is used and all of the information displayed by the **fts lsfldb** command.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine specified by **-server**.

Output

The **fts lsheader** command displays different output about the filesets at the specified location depending on whether the **-fast** or **-long** option is included. Information about the filesets is displayed in numeric order by fileset ID number if the **-fast** option is used; otherwise, it is displayed in alphabetical order by fileset name.

The following information is displayed for DCE Local File System filesets. Because non-Local File System filesets do not have DCE Local File System fileset headers, this command displays much less information for non-Local File System filesets, and the **-fast** and **-long** options have less of an impact on the amount of output displayed.

If the **-fast** option is used, the command lists the ID number of each fileset. If the **-aggregate** option is omitted, the command also displays the total number of filesets on the specified server.

If both the **-fast** and **-long** options are omitted, the command displays the following information:

- The File Server machine, aggregate name, and aggregate ID number where the filesets reside.
- The total number of filesets on the aggregate.
- Each fileset's name (with a **.readonly** or **.backup** extension, if appropriate).
- Each fileset's fileset ID number.
- Each fileset's type (RW for read-write, RO for read-only, or BK for backup.
- Each fileset's allocation usage and quota usage, in kilobytes.
- Each fileset's status (`On-line`, `Off-line`, or an error indicator).
- The total number of filesets on-line, the total number of filesets off-line, and the total number of filesets busy. A busy fileset is one upon which a fileset-related operation is currently in progress (for example, the fileset is being moved or cloned, or the Replication Server is currently forwarding changes from the fileset to read-only replicas).

If the **-long** option is used, the command displays the following additional information for each fileset:

- Whether it is a DCE Local File System or non-Local File System fileset.
- Information about the state of the fileset.
- The ID numbers of the parent, clone, and backup filesets related to the fileset.
- The ID numbers of the low-level backing and low-level forward filesets related to the fileset.
- The version number of the fileset.

- The allocation and allocation usage, in kilobytes, of the fileset.
- The quota and quota usage, in kilobytes, of the fileset.
- The day, date, and time when the fileset was created (replicated or backed up for a read-only or backup fileset).
- The day, date, and time when the contents of the fileset were last updated (same as the creation time for a read-only or backup fileset).

Examples

The following examples show output from the **fts lsheader** command when it is executed with the **-fast** option, with neither the **-fast** option nor the **-long** option, and with the **-long** option. All three examples display output primarily for the filesets: **charlieft1** (ID number **0,,352**), **charlieft2** (ID number **0,,354**), **charlieft3** (ID number **0,,358**), **charlieft4** (ID number **0,,361**), and **larryft** (ID number **0,,1070**).

```
$ fts lsheader dcefvt8 lfs8 -fast

0,,352
0,,354
0,,355
0,,357
0,,358
0,,360
0,,361
0,,363
0,,1070
0,,1072
```

```
$ fts lsheader dcefvt8 lfs8

Total filesets on server dcefvt8 aggregate lfs8 (id 8): 10
charlieft1             0,,352 RW     32 K alloc     37 K quota On-line
charlieft1.backup      0,,354 BK     37 K alloc     37 K quota On-line
charlieft2             0,,355 RW      8 K alloc     10 K quota On-line
charlieft2.backup      0,,357 BK     10 K alloc     10 K quota On-line
charlieft3             0,,358 RW      8 K alloc     10 K quota On-line
charlieft3.backup      0,,360 BK     10 K alloc     10 K quota On-line
charlieft4             0,,361 RW      8 K alloc     10 K quota On-line
charlieft4.backup      0,,363 BK     10 K alloc     10 K quota On-line
larryft                0,,1070 RW     24 K alloc    568 K quota On-line
larryft.backup         0,,1072 BK    568 K alloc    568 K quota On-line
Total filesets on-line 10; total off-line 0; total busy 0
```

```
$ fts lsheader dcefvt8 lfs8 -long

Total filesets on server dcefvt8 aggregate lfs8 (id 8): 10
charlieft1 0,,352 RW LFS     states 0x10010005 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,0      Clone 0,,353     Backup 0,,354
    llBack 0,,354    llFwd 0,,0       Version 0,,708
    4294967232 K alloc limit;        32 K alloc usage
          5000 K quota limit;        37 K quota usage
    Creation Mon Mar 25 17:14:33 1996
    Last Update Mon Mar 25 17:14:36 1996
```

```
charlieft1.backup 0,,354 BK LFS     states 0x10030006 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
     Parent 0,,352     Clone 0,,353     Backup 0,,354
     llBack 0,,0       llFwd 0,,352     Version 0,,708
     4294967232 K alloc limit;         37 K alloc usage
           5000 K quota limit;         37 K quota usage
     Creation Thu Mar  7 15:47:30 1996
     Last Update Mon Apr 29 16:53:44 1996

charlieft2 0,,355 RW LFS     states 0x10010005 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,0       Clone 0,,356     Backup 0,,357
    llBack 0,,357     llFwd 0,,0       Version 0,,6
    4294967232 K alloc limit;          8 K alloc usage
          5000 K quota limit;         10 K quota usage
    Creation Mon Mar 25 17:14:41 1996
    Last Update Mon Mar 25 17:14:41 1996

charlieft2.backup 0,,357 BK LFS     states 0x10030006 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,355     Clone 0,,356     Backup 0,,357
    llBack 0,,0       llFwd 0,,355     Version 0,,6
    4294967232 K alloc limit;         10 K alloc usage
          5000 K quota limit;         10 K quota usage
    Creation Thu Mar  7 15:47:44 1996
    Last Update Mon Apr 29 16:53:49 1996

charlieft3 0,,358 RW LFS     states 0x10010005 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,0       Clone 0,,359     Backup 0,,360
    llBack 0,,360     llFwd 0,,0       Version 0,,6
    4294967232 K alloc limit;          8 K alloc usage
         5000 K quota limit;         10 K quota usage
    Creation Mon Mar 25 17:14:38 1996
    Last Update Mon Mar 25 17:14:39 1996

charlieft3.backup 0,,360 BK LFS     states 0x10030006 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,358     Clone 0,,359     Backup 0,,360
    llBack 0,,0       llFwd 0,,358     Version 0,,6
    4294967232 K alloc limit;         10 K alloc usage
          5000 K quota limit;         10 K quota usage
    Creation Thu Mar  7 15:47:57 1996
    Last Update Mon Apr 29 16:53:47 1996

charlieft4 0,,361 RW LFS     states 0x10010005 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,0       Clone 0,,362     Backup 0,,363
    llBack 0,,363     llFwd 0,,0       Version 0,,6
    4294967232 K alloc limit;          8 K alloc usage
          5000 K quota limit;         10 K quota usage
    Creation Mon Mar 25 17:14:30 1996
    Last Update Mon Mar 25 17:14:30 1996
```

```
charlieft4.backup 0,,363 BK LFS     states 0x10030006 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,361     Clone 0,,362     Backup 0,,363
    llBack 0,,0       llFwd 0,,361     Version 0,,6
    4294967232 K alloc limit;         10 K alloc usage
         5000 K quota limit;         10 K quota usage
    Creation Thu Mar  7 15:48:13 1996
    Last Update Mon Apr 29 16:53:51 1996

larryft 0,,1070 RW LFS     states 0x10010005 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,0       Clone 0,,1071    Backup 0,,1072
    llBack 0,,1072    llFwd 0,,0       Version 0,,447
    4294967232 K alloc limit;         24 K alloc usage
          5000 K quota limit;        568 K quota usage
    Creation Sun Apr 21 15:26:12 1996
    Last Update Fri Apr 26 10:28:58 1996

larryft.backup 0,,1072 BK LFS     states 0x10030006 On-line
    dcefvt8.endicott.ibm.com, aggregate lfs8 (ID 8)
    Parent 0,,1070    Clone 0,,1071    Backup 0,,1072
    llBack 0,,0       llFwd 0,,1070    Version 0,,446
    4294967232 K alloc limit;        568 K alloc usage
          5000 K quota limit;        568 K quota usage
    Creation Tue Apr 16 17:45:47 1996
    Last Update Sun Apr 21 15:26:14 1996

Total filesets on-line 10; total off-line 0; total busy 0
```

Related Information

Commands:

fts lsfldb	**fts lsft**

File:

dfstab

fts lsmount

Purpose

Lists the filesets associated with mount points.

Format

fts lsmount -dir *directory_name*... **[-help]**

Options

-dir *directory_name*
: Names each directory that serves as a mount point for a fileset. The last element in the specified pathname must be an actual name, not . (dot) or .. (dot dot).

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsmount** command displays the name of the fileset for which each directory specified with the **-dir** option is the mount point. The association between a mount point and a fileset is created with the **fts crmount** command; it is removed with the **fts delmount** command.

Privilege Required

The issuer's DCE principal or DCE group must have read permission on each directory with the **-dir** option, regardless of whether each indicated directory resides in a directory in a DCE Local File System or non-Local File System fileset.

Output

The **fts lsmount** command displays the following message for each directory that is a mount point:

'directory_name' `is a mount point for fileset` *'fileset_name'*

In the output, *directory_name* is the name of a directory specified with the **-dir** option; *fileset_name* is the name of the fileset for which *directory_name* serves as a mount point. The command also provides the following information about the directory and fileset:

# (number sign)
: Precedes *fileset_name* if *directory_name* is a regular mount point

% (percent sign)
: Precedes *fileset_name* if *directory_name* is a read-write mount point

! (exclamation point)
: Replaces *fileset_name* if the directory is a global root mount point (a mount point for the root of the DCE global namespace)

The **fts lsmount** command displays the following message for each directory that is not a mount point:

'directory_name' `is not a mount point.`

Examples

The following example lists the mount point **vijay**, which is a regular mount point for the fileset named **user.vijay**:

```
$ fts lsmount vijay

'vijay' is a mount point for fileset '#user.vijay'
```

Related Information

Commands:

fts crmount	**fts delmount**

fts lsquota

Purpose

Shows quota and usage information about filesets and aggregates.

Format

fts lsquota [{-path {*filename* **|** *directory_name***}...| -fileset {***name* **|** *ID***}...}]**
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-path *filename* **|** *directory_name*
: Names a file or directory from each fileset about which quota and usage information is to be displayed. Include file names or directory names from different filesets if desired. It is not necessary to name more than one file or directory from the same fileset. Use this option or use **-fileset**; omit both options to display information about the fileset containing the current working directory.

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of each fileset about which quota and usage information is to be displayed. Use this option or use **-path**; omit both options to display information about the fileset that contains the current working directory.

-cell *cellname*
: Specifies the cell with respect to which the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. Generally, the **-noauth** option is included if DFS authorization checking is disabled on a server machine on which administrative privilege is required or if the Security service is unavailable. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions during command execution.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsquota** command displays quota and usage information about filesets and disk size. The command also provides usage information about filesets and the partitions or aggregates on which the filesets reside. Use the **-path** option to specify a file or directory on a fileset to see information about that fileset; use the **-fileset** option to specify the name or ID number of a fileset to see information about that fileset.

Note: You can omit both options to see information about the fileset containing the current working directory.

For DCE Local File System filesets, the **fts lsquota** command displays the quota and quota use (in kilobytes), and the percentage of the quota in use. For both DCE Local File System and non-Local File System filesets, this command displays the name of the fileset, information about the number of available kilobytes on the aggregate or partition on which the fileset resides, the number of kilobytes in use on the aggregate or partition, and the percentage of the aggregate or partition in use. It also reports whether the device is a DCE Local File System aggregate or a non-Local File System partition.

The size of a non-Local File System fileset is equal to the size of the partition on which it resides. Therefore, the size and usage information displayed for the partition (non-Local File System aggregate) in the output of the **fts lsquota** command equals the quota and quota usage information of the fileset on the partition.

The **fts lsheader** and **fts lsft** commands can be used to display the quota of a DCE Local File System fileset. The **fts aggrinfo** command can be used to display the total disk space on an aggregate and the amount currently available.

By default, every newly created DCE Local File System fileset has a quota of 5000 kilobytes. The **fts setquota** command can be used to increase or decrease the quota of a DCE Local File System fileset. Because the quota of a DCE Local File System fileset does not represent the amount of physical data stored on the fileset, it can be larger than the size of the aggregate on which the fileset resides. Similarly, the combined quotas of all filesets on an aggregate can be larger than the size of the aggregate.

The quota of a non-Local File System fileset cannot be changed by way of DFS. (The **fts setquota** command works only with DCE Local File System filesets.)

Privilege Required

No privileges are required.

Output

This command displays the following information about each specified fileset:

- The name of the fileset.
- The quota, in kilobytes, of the fileset (DCE Local File System only).
- The number of kilobytes of the quota currently in use on the fileset (DCE Local File System only).
- The percentage of the quota currently in use on the fileset (DCE Local File System only).
- The percentage of available disk space currently in use on the aggregate or partition on which the fileset resides.
- The number of kilobytes of disk space in use and available on the aggregate and the total number of kilobytes on the aggregate or partition on which the fileset resides.
- The file system type of the aggregate (Local File System or non-Local File System).

If the fileset quota usage rises above 90% or the aggregate or partition usage rises above 97%, the appropriate percentage is indicated with << and the message <<**WARNING** is displayed after the aggregate usage information at the end of the output line.

Note: Because each non-Local File System aggregate (partition) contains a single fileset, the information displayed for a non-Local File System aggregate applies to the single non-Local File System fileset it houses.

Examples

The command that follows lists quota and usage information for the filesets that contains the directory named **/.../abc.com/fs/usr/terry**. It also displays the size and usage information for the aggregate that contains this fileset. The command also displays size and usage information for the partition that contains the directory named **/.../abc.com/fs/usr/jlw**. The first directory resides on the DCE Local File System fileset named **user.terry**; the quota of the DCE Local File System fileset is less than the size of the aggregate on which it is located. The second directory resides on the non-Local File System fileset named **user.jlw**; the quota of the non-Local File System fileset is the same as the size of the partition on which it is located.

```
$ fts lsq /.../abc.com/fs/usr/terry /.../abc.com/fs/usr/jlw

Fileset Name   Quota   Used   % Used   Aggregate
user.terry     15000   5071      34%     86% = 84538/98300 (LFS)
user.jlw       10000   8448      84%     84% = 8448/10000 (non-LFS)
```

The following command lists quota and usage information for the DCE Local File System fileset named **user.jean**, and size and usage information for the aggregate on which the fileset resides. The <<**WARNING** message directs the issuer's attention to the fact that the percentage of the quota in use on the indicated fileset is well above the warning level of 90%.

```
$ fts lsq -f user.jean

Fileset Name   Quota   Used   % Used   Aggregate
user.jean       5000   4955      99%<<   92% = 87436/98300 (LFS) <<WARNING
```

When the **fts lsquota** command is issued against an RFS fileset, the values displayed in the following example are always used. This occurs because the DFS file server cannot accurately calculate OS/390 data set space utilization.

```
$ fts lsq -f rfsset

Fileset Name   Quota   Used   % Used   Aggregate
rfsset        120000  60000      50%   50% = 60000/120000 (non-LFS)
```

Related Information

Commands:

fts aggrinfo	**fts lsheader**	**fts setquota**
fts lsft		

fts lsreplicas

Purpose

Displays the status of DCE Local File System fileset replicas.

Format

fts lsreplicas -fileset {*name* **|** *ID***} {-all | -server** *machine***} [-cell** *cellname***]**
[{-noauth | -localauth}] [-verbose] [-help]

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the fileset whose replicas are to be checked.

-all
: Specifies that all replicas of **-fileset** are to be checked, regardless of where they are located. Use this option or use **-server**.

-server
: Names a specific File Server machine on which replicas of **-fileset** are to be checked. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option or use **-all**.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsreplicas** command shows the replication statuses of read-only replicas of the read-write DCE Local File System fileset specified with the **-fileset** option. Use the command's options to check replicas of **-fileset** as follows:

- To check the status of the replica stored on a specific File Server machine, specify the name of the machine with the **-server** option.
- To check the status of all replicas, specify the **-all** option.

If Release Replication is used for a read-write fileset, use the **fts release** command to place replicas of the fileset at replication sites. (If Scheduled Replication is used, the Replication Server automatically places replicas at replication sites according to specified parameters.) Use the **fts update** command to request that the Replication Server make an immediate update of the replicas of any read-write fileset.

Use the **fts statrepserver** command to check the status of the Replication Server process on a specific File Server machine. Use the **fts addsite** command to add a replication site; use the **fts rmsite** command to remove a replication site.

Privilege Required

No privileges are required.

Examples

The following command displays the status of each replica of the read-write fileset named **testft.bin** stored on server **dcefvt1**:

```
$ fts lsreplicas -fileset testft -s dcefvt1

testft, cell 268458673,,1821581880: src 0,,30 (LFS1) (on dcefvt1.endicott.ibm.co
om) => dcefvt1.endicott.ibm.com 0,,31 (LFS1)
  flags 0x20001, volstates 0x10423206.  NumKAs 0; lastKA sweep=Wed Dec 31 19:00:00 1969
   srcVV: 0,,1009; curVV: 0,,1009; WVT ID = 800573278,,283
  Lost token 7939227 ago; token expires 2700 hence; new version published 800736720 ago
  vvCurr 792797493.709762 (7939227 ago); vvPingCurr 800732220.024472 (4500 ago)
  Last update attempt 0.000000 (800736720 ago); next scheduled attempt 0.036824
 (-800736720 hence)
  Status msg: GetToken returned 0x404 token, ID 800573278,,283.
```

Related Information

Commands:

fts addsite	**fts rmsite**	**fts update**
fts release	**fts statrepserver**	

fts lsserverentry

Purpose

Lists a server entry from the FLDB.

Format

fts lsserverentry {-server *machine* **| -all} [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Specifies the name of the server machine whose entry in the Fileset Location Database (FLDB) is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option or use the **-all** option.

-all
: Specifies that the entries for all server machines in the FLDB are to be displayed. Use this option or use the **-server** option.

-cell *cellname*
: Specifies the cell from whose FLDB the specified server entries are to be listed. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts lsserverentry** command displays server entry information from the FLDB. If the **-server** option is specified, entry information from the FLDB for only the indicated server machine is displayed. If the **-all** option is specified, entry information from the FLDB for all server machines is displayed.

Use the **fts crserverentry** command to create a server entry in the FLDB. Use the **fts edserverentry** command to modify a server entry in the FLDB. Use the **fts delserverentry** command to remove a server entry from the FLDB.

Note: 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 know for that server.

Privilege Required

No privileges are required.

Examples

The following command displays the server entry from the FLDB for a server machine named **dcefvt1**:

```
$ fts lsserverentry -s dcefvt1

Description for site 'dcefvt1':
dcefvt1.endicott.ibm.com (2:0.0.9.130.79.35)
FLDB quota: 0;  uses: 35;  principal='hosts/DCEFVT1';  owner=<nil>
```

Related Information

Commands:

fts crserverentry	**fts delserverentry**	**fts edserverentry**

fts move

Purpose

Moves a read-write DCE Local File System fileset to another site.

Format

fts move -fileset {*name* **|** *ID***} -fromserver** *source_machine* **-fromaggregate** *source_name*
-toserver *dest_machine* **-toaggregate** *dest_name*
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or the fileset ID number of a read-write fileset to be moved.

-fromserver *source_machine*
: Names the File Server machine on which the fileset currently resides. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-fromaggregate *source_name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate on which the fileset currently resides. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-toserver *dest_machine*
: 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.

-toaggregate *dest_name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate to which the fileset is to be moved. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **dfstab** file.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts move** command moves the indicated read-write DCE Local File System fileset from its current site (specified with **-fromserver** and **-fromaggregate**) to the destination site (specified with the **-toserver** and **-toaggregate** options). The command decrements the number of fileset entries recorded as residing on the machine indicated with the **-fromserver** option in the Fileset Location Database (FLDB) entry for the machine, and it increments the number of fileset entries recorded as residing on the machine specified with the **-toserver** option in the FLDB entry for that machine. It also automatically removes the backup copy of the fileset, if it exists, from the current site. To create a new backup at the destination site, use the **fts clone** command.

If the fileset to be moved uses Scheduled Replication, the command has no effect on the fileset's replicas. However, if the fileset to be moved uses Release Replication, the command has the following effects:

- If the fileset is moved to a different File Server machine, the command removes the replication site and replica that resides at the fileset's current site. It then creates a new replication site and replica on the File Server machine and aggregate to which the fileset is moved. The new replica is created as a clone of the read/write fileset. If the machine to which the fileset is to be moved already houses a replication site and replica of the fileset, you must use the **fts rmsite** command to remove the existing replication site and replica from the destination machine before issuing the **fts move** command.
- If the fileset is moved to a different aggregate on the same File Server machine, the command does one of the following:
  - If the replica resides on the aggregate from which the fileset is moved, the command deletes the existing replication site and replica. It then creates a new replication site and replica on the aggregate to which the fileset is moved, creating the new replica as a clone.
  - If the replica resides on the aggregate to which the fileset is moved, the command has no immediate effect on the replica. However, the next time the **fts release** command is used, the replica is changed from a full read-only replica to a clone.
  - If the replica resides on neither of the aggregates involved in the move, the command does not affect the replica.

It is not possible to move a read-only or backup fileset. For read-only filesets, the corresponding action is to create a new replication site with the **fts addsite** command and remove an existing one with the **fts rmsite** command. Because the backup version of a read-write fileset is automatically deleted when its read-write source is moved, a backup fileset can be moved only by moving its read-write source and issuing the **fts clone** command to create a new backup version.

Furthermore, it is not possible to move a fileset from a site in one cell to a site in another cell. Filesets can be moved only between two sites in the same cell. The filesets are assumed to reside in the local cell of the issuer unless the name of a foreign cell is specified with the **-cell** option.

A DCE Local File System fileset that is mounted locally (as a file system on its File Server machine) cannot be moved to a different File Server machine. It can be moved only to a different aggregate on the same File Server machine. If the command is used to move a DCE Local File System fileset that is locally mounted, its **-fromserver** and **-toserver** options must name the same File Server machine; otherwise, the command is not successful. (To move a locally mounted fileset to a different server machine, remove its local mount point before issuing this command.)

Note: You cannot locally mount a DCE Local File System fileset on OS/390 DFS.

In addition, because the backup version of a fileset is removed when its read-write version is moved, you cannot move a fileset (not even to another aggregate on the same File Server machine) if its backup version is mounted locally. You must remove the backup version's local mount point before moving the fileset.

Note: If you are moving a DCE Local File System fileset from an OS/390 system to an OS/390 system, it is recommended that you issue the **fts move** command from an OS/390 system.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** files on both the source and destination machines. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entries for the source machine, the destination machine, and any machines on which replicas of the fileset reside. In addition, the source machine (**-fromserver**) must be listed in the **admin.ft** file on the destination machine (**-toserver**).

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Examples

The following command moves the fileset **user.smith** from **/dev/ufs1** on **fs3** to **/dev/ufs2** on **fs7**:

```
$ fts move user.smith /.../abc.com/hosts/fs3 /dev/ufs1 /.../abc.com/hosts/fs7 /dev/ufs2
```

Implementation Specifics

You cannot locally mount a DCE Local File System fileset on OS/390 DFS.

Related Information

Commands:

fts addsite	**fts delete**	**fts rmsite**
fts clone	**fts release**	

File:
dfstab

fts release

Purpose

Initiates Release Replication by placing a read-only version of a read-write DCE Local File System fileset at the local site.

Format

fts release -fileset {*name* **|** *ID***} [-wait] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the read-write fileset to be replicated locally (cloned if the local replication site is defined on the same aggregate as the read-write fileset). Once the fileset is replicated locally, the Replication Servers at the fileset's replication sites copy the replica to their sites.

-wait
: Directs **fts** to wait until the Replication Servers have copied the latest replica to their sites before completing execution. By default, the command returns immediately without waiting for the Replication Servers to complete propagation.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts release** command is used to initiate the replication process for a fileset that uses Release Replication. The command "releases" a new read-only copy of the DCE Local File System fileset specified with the **-fileset** option. It places a new read-only copy at the local replication site defined on the same File Server machine as the read-write fileset. The Replication Servers at each of the fileset's replication sites (specified File Server machines and aggregates) then place read-only replicas of the copy at the sites on their respective machines.

Note that, as with updating a new version of a fileset that uses Scheduled Replication, releasing a fileset that uses Release Replication does not ensure immediate access to data in the new version of the replica. A Cache Manager may continue to provide data cached from an old version of a read-only replica until the expiration of the MaxAge for the fileset containing that data or until the Cache Manager needs to access data from the replica that it has not already cached.

To gain immediate access to data in the new version of the replica, issue the the **cm flush** or the **cm flushfileset** command to flush the old data from the cache. This forces the Cache Manager to replace data it has cached from the replica. However, because Replication Servers update replicas one at a time, the Cache Manager can still provide data from the old version of the replica if it accesses a replica that has yet to be updated. Until all replicas have been updated, you cannot directly force the Cache Manager to access data from the new version of the replica.

Before the **fts release** command can be used, the **fts setrepinfo** command must be used to define the replication parameters for the read-write fileset. If Release Replication is to be used, the **-release** option must be specified with the **fts setrepinfo** command. The **fts addsite** command must also be used to define the replication sites for the read-write fileset. For Release Replication, the replication site on the same File Server machine as the read-write fileset must be defined first. The read-write fileset must have at least one replication site defined before the **fts release** command can be issued. The replication parameters and sites for a read-write fileset are recorded in the fileset's entry in the Fileset Location Database (FLDB).

The **fts release** command does not alter the replication type and parameters defined for the specified fileset. The command can be used only with a fileset that uses Release Replication: it returns an error if the specified fileset uses Scheduled Replication. The **fts update** command can be used to request an immediate update of the replicas of a fileset that uses Scheduled Replication.

Use the **fts lsreplicas** command to check the status of replicas. Use the **fts statrepserver** command to check the status of the Replication Server on a File Server machine.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine on which the source read-write fileset is stored. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entries for the machine on which the source fileset resides and all machines on which the read-only replicas are to reside.

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Examples

The following command initiates Release Replication for the read-write fileset named **pmax_osf1.bin**:

```
$ fts release pmax_osf1.bin
```

Related Information

Commands:

cm flush	**fts lsreplicas**	**fts statrepserver**
cm flushfileset	**fts setrepinfo**	**fts update**
fts addsite		

fts rename

Purpose

Renames a fileset.

Format

fts rename -oldname *oldname* **-newname** *newname* **[-cell** *cellname*] **[{-noauth | -localauth}]**
[-verbose] [-help]

Options

-oldname *oldname*
Specifies the current name of the read-write fileset.

-newname *newname*
Specifies the new name for the read-write fileset. The name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. It can be no longer than 102 characters. This length does not include the **.readonly** or **.backup** extension, which is added automatically when a read-only or backup version of the fileset is created. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup DCE Local File System filesets, so you cannot specify a fileset name that ends with either of those extensions.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts rename** command changes the name of the read-write fileset specified with **-oldname** to the name specified with **-newname**. The names of the read-write fileset's read-only copies and backup copy, if any, automatically change to match.

After issuing this command, the issuer must correct any mount points that refer to the old fileset name. This is done by removing each old mount point with the **fts delmount** command and creating a replacement for each with the **fts crmount** command. (These commands require that the issuer have the necessary file system permissions for the operations.)

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine on which the read-write fileset resides. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be renamed resides.

Examples

The following command changes the incorrect fileset name **osf1.bin** to the correct fileset name **pmax_osf1.bin**:

```
$ fts rename osf1.bin pmax_osf1.bin
```

Related Information

Commands:

fts crmount	**fts delmount**

fts restore

Purpose

Converts a dump file from byte stream format to fileset format and places it in the file system.

Format

fts restore -ftname *name* **-server** *machine* **-aggregate** *name* **[-file** *filename***] [-ftid** *ID***]**
[-overwrite] [-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-ftname *name*
Specifies the name of the fileset to which the file is to be restored. If the file is to be restored as a new fileset, the name must be unique within the local cell, and it should be indicative of the fileset's contents. The following characters can be included in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a to z, and A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name must contain at least one alphabetic character or an _ (underscore) to differentiate it from an ID number. It can be no longer than 102 characters. This length does not include the **.readonly** or **.backup** extension, which is added automatically when a read-only or backup version of the fileset is created. Note that the **.readonly** and **.backup** extensions are reserved for use with read-only and backup DCE Local File System filesets, so you cannot specify a fileset name that ends with either of those extensions.

-server *machine*
Specifies the File Server machine to which the file is to be restored. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** to which the file is to be restored. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file.

-file *filename*
Specifies the complete pathname of the file to be restored. If a complete pathname is not provided, the file is assumed to reside in the current working directory. If this option is omitted, the data is read from standard input (**STDIN**).

Note: The **-file** option on OS/390 DFS refers to an OS/390 HFS file.

-ftid *ID*
Specifies the fileset ID number to assign to the restored fileset. If this option is omitted and an existing fileset is to be overwritten, the fileset ID number of the existing fileset is used. If it is omitted and a new fileset is to be created, the FL Server allocates a new fileset ID number for the fileset. Use this option only when restoring a dump file as a DCE Local File System fileset; use it sparingly and with great care. Omit this option when restoring a dump file as a non-Local File System fileset.

Important Note to Users

On OS/390 DFS, the **-ftid** option is ignored.

-overwrite
Specifies that the file to be restored can overwrite an existing fileset. If this option is omitted, the command exits without overwriting an existing fileset. You must use this option to overwrite a previously restored version of a fileset with an incremental dump of the same fileset; more information about conditions that must be met if a fileset is to be overwritten by an incremental dump is provided in “Usage.” You must also use this option to restore a dump file as a non-Local File System fileset.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts restore** command translates a dump file created previously with the **fts dump** command from a byte stream format to a fileset format appropriate for the machine specified with the **-server** option. The dump file to be restored is indicated with the **-file** option. If this option is omitted, the data to be restored is read from **STDIN**.

The fileset contained in the dump file can be restored as a new read-write DCE Local File System fileset by specifying a new name and site for the fileset. The command assigns the fileset the name indicated with the **-ftname** option. It restores it to the site specified with the **-server** and **-aggregate** options. The dump file must contain the full dump of a fileset if it is to be restored as a new fileset.

Alternatively, the fileset contained in the dump file can be restored over an existing read-write version of the same fileset by specifying the name and site of the existing fileset. The command resets the creation time stored in the fileset's header to match the restore time. The **-overwrite** option must be used to specify that the dump file is to overwrite the existing fileset. If this option is omitted, the command displays an error message and exits instead of overwriting the existing fileset.

When restoring a dump file as a non-Local File System fileset, the fileset must already exist for the non-Local File System partition on which it resides to be exported to the DCE namespace. In this case, you must use the **-overwrite** option to overwrite the existing non-Local File System fileset (even if the fileset to be overwritten contains no data).

Note: On OS/390 DFS, you cannot restore a non-Local File System fileset that resides on OS/390.

If you are overwriting an existing fileset with an incremental dump, the fileset to be overwritten should initially have been restored as a new read-write fileset from a full dump. Also, both the dump file to be restored and the full dump that initially produced the read-write fileset to be overwritten must be dumps of the same fileset. (A full dump of a fileset can be restored to overwrite an existing fileset, but the restored dump file overwrites all data in the existing fileset. An incremental dump of a fileset cannot be restored to overwrite an existing fileset that was not created from the restoration of a full dump.)

Multiple incremental dumps of a fileset can be restored to overwrite the same existing fileset provided the following conditions are true:

- The fileset to be overwritten must not have been modified (that is, no files added, removed, or saved, and no ACLs changed) since its most recent restoration from a full or incremental dump.
- The dump file to be restored must have been created *from* a date and time (as specified with the **-time** or **-version** option of the **fts dump** command) *no later* than the date and time at which the most recently restored dump of the fileset to be overwritten was dumped.
- The dump file to be restored must have been created at a date and time *later* than the date and time at which the most recently restored dump of the fileset to be overwritten was dumped.

The last two conditions indicate that the span of time recorded in the incremental dump to be restored must overlap and extend the span of time recorded in the fileset to be overwritten. For example, suppose the following dumps were made of a fileset: a full dump was made on 1 January 1992; an incremental dump from 31 December 1991 was made on 7 January 1992; and an incremental dump from 6 January 1992 was made on 14 January 1992. The following sequence of operations represents the only possible way to restore the fileset from all three of these dumps:

1. The full dump made on 1 January is restored as a new read-write fileset.
2. The incremental dump made on 7 January is restored to overwrite the read-write version of the fileset made from the full dump.
3. The incremental dump made on 14 January is restored to overwrite the read-write version of the fileset that includes data from the full and first incremental dumps.

No other sequence of restore operations involving all three dumps is possible. Any other sequence of steps will undoubtedly result in some or all of the data in the fileset being inaccessible or inconsistent.

When restoring a dump file as a DCE Local File System fileset, a fileset ID number can be assigned to the restored fileset with the **-ftid** option. This is generally not recommended unless there is good reason to believe that an available fileset ID number can be specified. If the **-ftid** option is omitted, an overwritten DCE Local File System fileset retains its current ID number, or the FL Server allocates a new ID number for a new DCE Local File System fileset restored from a dump file. If a new fileset ID number is assigned or allocated, the FL Server increments the number of fileset entries recorded as residing on the specified File Server machine in the Fileset Location Database (FLDB) entry for the server.

When restoring a dump file as a non-Local File System fileset, do not use the **-ftid** option. Omit the option to continue to use the fileset ID number specified for the non-Local File System fileset in the entry for its partition in the **dfstab** file. (Note that the restored dump file overwrites all data on the non-Local File System partition.)

Important Note to Users

On OS/390 DFS, the **-ftid** option is ignored.

If a new fileset is created, use the **fts crmount** command to create a mount point for the fileset, making it visible in the DCE namespace. If an existing DCE Local File System fileset is overwritten with this command, use the **fts update** command to release new read-only replicas based on the new version of the fileset, and use the **fts clone** command to create a new backup version of the fileset, as necessary.

You can use the **fts restore** command to restore a dump file to any type of fileset (DCE Local File System or non-Local File System), regardless of the type of fileset from which it was created. For example, a dump file of a DCE Local File System fileset can be restored to a DCE Local File System fileset or to any type of non-Local File System fileset. Similarly, a dump file of a non-Local File System fileset can be restored to a DCE Local File System fileset or to a different type of non-Local File System fileset. In any case, the contents of the dump file are translated into the appropriate format for the file system to which they are restored.

Note: On OS/390 DFS, you cannot restore a non-Local File System fileset that resides on OS/390.

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 Local File System fileset may be lost if the fileset is restored to a file system that does not support ACLs.

You cannot restore a fileset dumped in one cell to a site in another cell.

Cautions

Ensure that all of the conditions discussed in the description section are met before restoring an incremental dump of a fileset over an existing fileset. Violation of any of the conditions is very likely to result in inaccessibility or inconsistency of some or all of the data in the fileset.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine specified by **-server** and must have the read permission on the dump file. The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset is recorded as residing in the FLDB (generally only **-server** unless an existing fileset is to be overwritten).

The DCE principal or DCE group **must** also be added to the access control list for the OS/390 endpoint map.

Examples

The following example restores a file, **/tmp/smith.013191.dump**, that contains an incremental dump of a fileset over an existing read-write version of the same fileset, **user.smith**. The incremental dump was created using a start date and time no later than the date and time when the most recently restored version of the fileset to be overwritten was dumped, and it was dumped at a date and time later than the date and time when the most recently restored version of the fileset to be overwritten was dumped. Also, the fileset to be overwritten has not been modified since it was last restored. The **-ftid** option is omitted, so the fileset retains its current fileset ID number.

```
$ fts restore user.smith /.../abc.com/hosts/fs1 lfs1 /tmp/smith.013191.dump -overwrite
```

The following command takes input directly from an **fts dump** command to create a new read-write fileset, **user.terry**, from an existing fileset, **user.smith**. The **-file** option is omitted from the **fts dump** command to send the output to **STDOUT**, and it is omitted from the **fts restore** command to read the input from **STDIN**. (The information is "piped" from one command to the next.) The **-ftid** option is again omitted from the **fts restore** command; this time the FL Server allocates a new ID number for the fileset.

```
$ fts dump user.smith -time 0 | fts restore user.terry /.../abc.com/hosts/fs1 lfs1
```

Implementation Specifics

The **-file** option on OS/390 DFS refers to an OS/390 HFS file.

On OS/390 DFS, you cannot restore a non-Local File System fileset that resides on OS/390.

On OS/390 DFS, the **-ftid** option is ignored.

Related Information

Commands:
fts clone
fts crmount
fts dump
fts update

File:
dfstab

fts rmsite

Purpose

Removes a replication site and read-only DCE Local File System fileset.

Format

fts rmsite -fileset {*name* **|** *ID***} -server** *machine* **-aggregate** *name* **[-cell** *cellname***]**
[{-noauth | -localauth}] [-verbose] [-help]

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the read-write fileset for which a replication site and the read-only fileset stored at that site are to be removed.

-server *machine*
: Specifies the File Server machine to be removed as a replication site. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of the aggregate to be removed as a replication site. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file. If the aggregate is not currently exported or has been detached, you must specify the aggregate ID.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts rmsite** command removes a replication site currently defined for the read-write DCE Local File System fileset specified with the **-fileset** option. The **-server** and **-aggregate** options are used to specify the replication site to be removed. The command performs the following actions:

- It removes the definition of the replication site from the Fileset Location Database (FLDB) entry for the fileset.
- It decrements the number of fileset entries recorded as residing on the File Server machine specified with **-server** in the FLDB entry for the server.

- If the indicated fileset uses Release Replication and the specified site is on the same File Server machine as the read-write fileset, the command removes the replica (if it exists); see the "Cautions" section for more information. For any other replica, the command instructs the Replication Server at the site to remove the replica.

Other replication sites of the read-write fileset are not affected. If the command is used to remove a fileset's last replication site, the status flag for the read-only version in the fileset's FLDB entry is set to `invalid`. If it is used to remove the last existing version of a fileset, the fileset's entire FLDB entry is removed.

Before you use the **fts delete** command to remove the read-write (and backup) version of a fileset, use the **fts rmsite** command to remove the fileset's replication sites. If Release Replication was used for the fileset, use the **fts rmsite** command to remove the replication site (and replica) stored on the same File Server machine as the read-write fileset as well.

If the aggregate on which the replication site is defined is not currently exported or has been detached with the **dfsexport** command, you must specify the aggregate ID of the aggregate; otherwise, the **fts rmsite** command cannot remove the replication site. If the aggregate is not exported or has been detached, the Replication Server on the File Server machine on which the aggregate resides stops trying to maintain the replica at the site once the **fts rmsite** command is issued, and it removes the replica from the site once the aggregate is again exported.

Replication sites are added with the **fts addsite** command. The replication type for a read-write fileset is set or changed with the **fts setrepinfo** command.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of the fileset for which the replication site and replica are to be removed. The issuer's DCE principal or DCE group must also be listed in the **admin.ft** file on the machine specified by **-server** if the following are true:

- Release Replication is used for the fileset.
- The replication site on the same File Server machine as the read-write fileset is to be removed (in which case **-server** names the File Server machine on which the read-write fileset resides).
- A replica actually exists at the specified replication site.

Cautions

If you use Release Replication and you remove the read-only fileset that is on the same File Server machine as the read-write source, all other read-only filesets become unavailable upon the expiration of the fileset's FailAge parameter. The FailAge parameter is set using the **fts setrepinfo** command.

Examples

The following command removes the replication site on the aggregate **/dev/ufs1** of the File Server machine **fs5** from the FLDB entry for the fileset named **applications.bin**. A replica of **applications.bin** that resides at the site is also removed.

```
$ fts rmsite applications.bin /.../abc.com/hosts/fs5 /dev/ufs1
```

Related Information

Commands:

fts addsite	**fts delete**	**fts setrepinfo**

File:

dfstab

fts setprotectlevels

Purpose

Sets advisory DCE remote procedure call (RPC) authentication levels for a specified fileset.

Format

fts setprotectlevels -fileset {*name* **|** *ID***} [-minlocalprotectlevel** *level***] [-maxlocalprotectlevel** *level***] [-minremoteprotectlevel** *level***] [-maxremoteprotectlevel** *level***] [-cell** *cellname***] [-verbose] [-noauth | -localauth] [-help]**

Options

-fileset {*name* **|** *ID***}**
: Specifies a fileset either by its name or volume ID.

-minlocalprotectlevel *level*
: Specifies the advisory lower bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within the same cell). The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage" on page 692.

-maxlocalprotectlevel *level*
: Specifies the advisory upper bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within the same cell). The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage" on page 692.

-minremoteprotectlevel *level*
: Specifies the advisory lower bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within foreign cells). The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage" on page 692.

-maxremoteprotectlevel *level*
: Specifies the advisory upper bound DCE RPC authentication level for the specified fileset (used by DFS client Cache Managers within foreign cells). The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Usage" on page 692.

-cell *cellname*
: Specifies the cell as *cellname* within which the specified fileset resides.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts setprotectlevels** command adjusts the minimum and maximum advisory DCE RPC authentication level bounds for a specified fileset. These bounds are used to bias a Cache Manager to a higher or lower security level when accessing the specified fileset. However, the bounds are simply advisory in that if the Cache Manager's security level settings are outside of the advisory bounds, the Cache Manager can cross the advisory and continue negotiating with a File Server. In this case, the Cache Manager's minimum security level (set with the **dfsd** or **cm setprotectlevels** command) and the File Server's maximum security bound (set with the **fxd** command) become the "hard" limits. Note that if the **fts setprotectlevels** bounds fall outside of File Server bounds, the File Server bounds take precedence.

In practice, when a Cache Manager must access a given fileset it first consults a Fileset Location (FL) Server for the location of that fileset (or any replicas if it is replicated read-only fileset). Along with the location, the Cache Manager also receives the applicable minimum and maximum advisory bounds for that fileset. The Cache Manager then checks its initial authentication level and compares that to the range defined by the bounds. The Cache Manager then adjusts its initial authentication level as follows:

- If the Cache Manager's initial authentication level is within the range defined by the advisory bounds, the initial level is used without adjustment.
- If the Cache Manager's initial authentication level is above the maximum advisory bound, the Cache Manager adjusts the initial level to match the advisory upper bound. However, the Cache Manager will not adjust its authentication level below its own minimum setting.
- If the Cache Manager's initial authentication level is below the minimum advisory bound, the Cache Manager adjusts the initial level to match the advisory lower bound.

The negotiation process to set an RPC authentication level now occurs as usual between the Cache Manager and File Server. The Cache Manager sends an RPC using the initial authentication level (which may have been adjusted because of the advisory bounds) to the File Server. If the initial authentication level is outside the minimum or maximum bounds set at the File Server, the File Server returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Server requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Server). Once the Cache Manager and File Server have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Server.

Note that the use of this command does not preclude communication with Cache Managers running earlier versions of DCE.

The various authentication levels are set by specifying either an integer value between 0 and 7, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **0** or **default** or **rpc_c_protect_level_default**: Use the DCE default authentication level.
- **1** or **none** or **rpc_c_protect_level_none**: Perform no authentication.
- **2** or **connect** or **rpc_c_protect_level_connect**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **3** or **call** or **rpc_c_protect_level_call**: Authenticate only at the beginning of each RPC received.
- **4** or **pkt** or **rpc_c_protect_level_pkt**: Ensure that all data received is from the expected host.
- **5** or **pkt_integrity** or **rpc_c_protect_level_pkt_integrity**: Authenticate and verify that none of the data transferred has been modified.

- **6** or **pkt_privacy** or **rpc_c_protect_level_pkt_privacy**: Perform authentication as specified by all of the previous levels and also encrypt each RPC argument value.
- **7** or **cdmf_priv** or **rpc_c_protect_level_cdmf_priv**: Perform authentication as specified by all of the pervious levels (except 6) and also encrypt each RPC argument value. This level provides a lower level of packet privacy than **rpc_c_protect_pkt_privacy**.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

Privilege Required

The issuer must have FLDB administration privileges or must be in the owner group for the File Server.

Examples

The following command sets the following authentication values:

- The maximum advisory authentication level for communication with Cache Managers in the local cell is set to packet integrity.
- The minimum advisory authentication level for communication with Cache Managers in the local cell is set to packet.
- The maximum advisory authentication level for communication with Cache Managers in foreign cells is set to packet security.
- The minimum advisory authentication level for communication with Cache Managers in foreign cells is set to packet security.

```
$ fts setprotectlevels -fileset richland.12 -maxlocalprotectlevel 5 -minlocalprotectlevel 4 \
                       -maxremoteprotectlevel 6 -minremoteprotectlevel 6
```

Related Information

Commands:

cm getprotectlevels	**dfsd**	**fxd**
cm setprotectlevels		

There is no **fts getprotectlevels** command. You may use the **fts lsfldb** or **fts lsft** commands to list protection levels that have been set with **fts setprotectlevels**.

fts setquota

Purpose

Sets the maximum quota for a read-write DCE Local File System fileset.

Format

fts setquota {-path {*filename* **|** *directory_name***} | -fileset {***name* **|** *ID***}} -size** *kbytes*
[-cell *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-path *filename* **|** *directory_name*
: Names a directory or file located on the read-write fileset whose quota is to be set. Use this option or use **-fileset**.

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the read-write fileset whose quota is to be set. Use this option or use **-path**.

-size *kbytes*
: Specifies the maximum amount of disk space that all of the files and directories in the read-write fileset can occupy. This includes files and directories in the read-write version of the fileset that are actually pointers to disk blocks in the backup or read-only version of the fileset. Specify the value in 1-kilobyte blocks. (A value of 1024 kilobytes is 1 megabyte.) By default, every newly created fileset has a quota of 5000 kilobytes.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts setquota** command sets the quota limit for a read-write DCE Local File System fileset. (It cannot be used to set the quota for a non-Local File System fileset or for a read-only or backup DCE Local File System fileset.) The fileset whose quota is to be set can be indicated by specifying the name of a file or directory on the fileset with the **-path** option or by indicating the fileset directly with the **-fileset** option.

Quota refers to the amount of disk space occupied by all of the files and directories in the read-write version of the fileset. This includes files and directories in the read-write version of the fileset that are actually pointers to disk blocks in the backup or read-only version of the fileset. Do not confuse quota with allocation; the latter identifies the amount of disk space occupied by the data that a fileset actually houses; excluding those files and directories that are pointers to disk blocks in another version of the fileset.

By default, every newly created fileset has a quota of 5000 kilobytes. This command increases or decreases a fileset's quota to be the number of kilobytes specified with the **-size** option. Because it does not represent the amount of physical data the fileset contains, a fileset's quota can be larger than the size of the aggregate on which it resides. Similarly, the sum of the quotas of all filesets on an aggregate can exceed the size of the aggregate.

The **fts lsft**, **fts lsheader**, and **fts lsquota** commands display, among other things, the current quota for a fileset.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine on which the fileset is stored.

Examples

The following command sets the quota for the fileset named **user.terry** to be 15,000 kilobytes:

```
$ fts setq -fileset user.terry -size 15000
```

Related Information

Commands:

fts lsft	**fts lsheader**	**fts lsquota**

fts setrepinfo

Purpose

Sets or changes replication type and parameters for a read-write DCE Local File System fileset.

Format

fts setrepinfo -fileset {*name* **|** *ID***} [{-release | -scheduled}] [-change] [-maxage** *interval***] [-failage** *interval***] [-reclaimwait** *interval***] [-minrepdelay** *interval***] [-defaultsiteage** *interval***] [-clear] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the read-write source fileset for which the replication type and parameters are to be set or changed. This command is used to set parameters for either Release or Scheduled Replication.

-release
: Specifies that Release Replication is to be used with the fileset indicated with the **-fileset** option. When initially defining a fileset's replication parameters, use this option or use **-scheduled**. Afterward, omit both options when modifying the fileset's replication parameters without changing its replication type.

To change a fileset's replication type (from Release to Scheduled, or from Scheduled to Release), include both the **-change** option and either the **-release** or **-scheduled** option to indicate the new type of replication to be used with the fileset.

-scheduled
: Specifies that Scheduled Replication is to be used with the fileset indicated with the **-fileset** option. When initially defining a fileset's replication parameters, use this option or use **-release**. Afterward, omit both options when modifying the fileset's replication parameters without changing its replication type.

To change a fileset's replication type (from Release to Scheduled, or from Scheduled to Release), include both the **-change** option and either the **-release** or **-scheduled** option to indicate the new type of replication to be used with the fileset.

-change
: Specifies that the type of replication currently used with the fileset indicated with the **-fileset** option is to be changed. Include the **-release** option to change the fileset's replication type from Scheduled to Release; include the **-scheduled** option to change the fileset's replication type from Release to Scheduled. Omit this option when specifying **-release** or **-scheduled** to initially set a fileset's replication type. Also omit it when changing a fileset's replication parameters without changing its replication type.

-maxage *interval*
: Specifies the amount of time the Cache Manager distributes data cached from a read-only replica without attempting to verify that the data is current. The Replication Server maintains information about the currentness of a read-only replica, which it communicates to the Cache Manager by way of the File Exporter. For Scheduled Replication, a replica must remain current with respect to the read-write fileset; for Release Replication, a replica must remain current with respect to the read-only fileset that resides on the same File Server machine as the read-write source fileset. The default is 2 hours. An effective value must be greater than or equal to 2 minutes.

This option is applicable to Release and Scheduled Replication.

-failage *interval*
Specifies the amount of time the Cache Manager distributes data cached from a read-only replica if that data cannot be verified as current. The difference between FailAge and MaxAge is the amount of time the Cache Manager continues to distribute data cached from a read-only replica after that data cannot be verified as current. The default is 1 day or twice the MaxAge, whichever is larger. An effective value must be greater than or equal to **-maxage**. *Applicable to Release and Scheduled Replication.*

This option is applicable to Release and Scheduled Replication.

-reclaimwait *interval*
Specifies the amount of time the File Exporter waits before it reclaims storage space from deleted files; that is those not referenced by any directory (ReclaimWait). It also determines the frequency of the Cache Manager's keep-alive messages to the Replication Server.

The Cache Manager sends keep-alive messages to indicate that it is still using files on a read-only replica. A file being accessed from a replica remains available as long as the Cache Manager continues to notify the Replication Server that the file is still in use and the Replication Server continues to forward these notifications to the File Exporter. This is true even if the file has been removed from all directories on the read-write fileset in the interim. The Cache Manager sends keep-alive messages more frequently than the ReclaimWait interval to prevent the File Exporter from reclaiming storage space occupied by deleted files. The default is 18 hours. An effective value must be greater than 2 hours; do not specify a value less than 90 minutes.

This option is applicable to Release and Scheduled Replication.

-minrepdelay *interval*
Specifies how long the Replication Server waits after a read-write source fileset changes before it attempts to get a new copy of the fileset (MinRepDelay). The Replication Server tracks the currency of replicas by maintaining a whole-fileset token for each fileset. If a Cache Manager changes the read-write fileset, the Replication Server relinquishes its whole-fileset token and waits for at least the time specified by MinRepDelay before requesting a new whole-fileset token. The default is 5 minutes or one-quarter of the DefaultSiteAge, whichever is smaller. This value must be less than the MaxSiteAge specified for each replication site with the **-maxsiteage** option of the **fts addsite** command.

This option is applicable to Scheduled Replication only.

-defaultsiteage *interval*
Specifies the default value to be used as the MaxSiteAge for a replication site (DefaultSiteAge). The DefaultSiteAge is used if the **-maxsiteage** option is omitted when the **fts addsite** command is used to add a replication site. The default is one-quarter of the MaxAge.

This option is applicable to Scheduled Replication only.

-clear
Removes all replication parameters previously defined for the fileset. The options associated with the type of replication in use for the fileset can then be used to define new replication parameters, or they can all be omitted to allow the system to calculate new replication parameters for the fileset.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts setrepinfo** command is used to set or change the replication type and parameters for a read-write DCE Local File System fileset. It affects the parameters for both Release and Scheduled Replication. It must be issued before replication sites can be defined for the fileset with the **fts addsite** command and before the **fts release** or **fts update** command can be used to copy replicas to the replication sites. The replication type and parameters for a fileset are stored in the fileset's entry in the Fileset Location Database (FLDB).

Use the following guidelines when deciding which type of replication (Release or Scheduled) to use with a read-write fileset:

- Use Release Replication if the fileset seldom changes or if the distribution of replicas must be closely tracked.
- Use Scheduled Replication if having the system release replicas of the fileset at regular intervals is preferred and the distribution of replicas does not need to be tracked.

When initially defining a fileset's replication type, include either the **-release** or **-scheduled** option. These options are then omitted from the command unless the replication type for the fileset is being changed (from Release to Scheduled, or from Scheduled to Release). To change the replication type, use the appropriate option (**-release** or **-scheduled**) to specify the new type, and include the **-change** option to indicate that the type is to be changed.

Note that, because Release Replication does not require a replication site to have a MaxSiteAge, it is likely that one or more Release Replication sites will have a MaxSiteAge of 0 (zero), which is the default value recorded for a site if no MaxSiteAge or DefaultSiteAge is specified. When changing from Release Replication to Scheduled Replication, the **-defaultsiteage** option *must be* used to set a DefaultSiteAge if any replication site does not have a MaxSiteAge and no DefaultSiteAge exists for the source fileset; otherwise, the **fts setrepinfo** command fails. If the command fails for this reason, reissue it, specifying a DefaultSiteAge with the **-defaultsiteage** parameter.

The **-maxage**, **-failage**, **-reclaimwait**, **-minrepdelay**, and **-defaultsiteage** options are used to set the corresponding replication parameters for a read-write fileset (see the **Options** section for information on the replication parameter each option sets). The following table lists each option's default value and describes the dependencies between the different options when they are used to set the replication parameters for either Release or Scheduled Replication.

Table 18. Replication Parameters

Parameter	Default	Release Replication	Scheduled Replication
-maxage	2 hours	*Required only if* **-failage** is specified.	*Required only if* one of the following is specified: **-failage**, **-minrepdelay**, or **-defaultsiteage**.
-failage	The larger of 1 day or twice **-maxage**	*Optional.* If it is specified, the following are required: **-maxage** and **-reclaimwait**.	*Required only if* one of the following is specified: **-minrepdelay** or **-defaultsiteage**.
-reclaimwait	18 hours	*Required only if* **-failage** is specified.	*Required only if* one of the following is specified: **-failage**, **-minrepdelay**, or **-defaultsiteage**.
-minrepdelay	The smaller of 5 minutes or one-quarter of **-defaultsiteage**	*Not applicable.*	*Required only if* one of the following is specified: **-failage** or **-defaultsiteage**.
-defaultsiteage	One-quarter of **-maxage**	*Not applicable.*	*Optional.* But if the other options are specified and **-defaultsiteage** is not, the **-maxsiteage** option of the **fts addsite** command is required when defining replication sites for the fileset.

The **fts** program calculates default values for each of the parameters *unless*

- **-failage** is specified for Release Replication.
- **-failage**, **-minrepdelay**, or **-defaultsiteage** is specified for Scheduled Replication.

Once one of these options is specified, the **fts** program no longer performs any default calculations; *interval* must be provided for all applicable options. (The exception is **-defaultsiteage** for Scheduled Replication, which is always optional.) Also, because the **-minrepdelay** and **-defaultsiteage** options do not apply to Release Replication, they are recorded if specified but they are ignored.

Enter *interval* values as integers, using the following abbreviations to indicate units: **d** for days, **h** for hours, **m** for minutes, and **s** for seconds. The syntax for an *interval* is

```
[integer d] [integer h] [integer m] [integer s]
```

At least one of the four values (days, hours, minutes, or seconds) must be provided, and a unit abbreviation (**d**, **h**, **m**, or **s**) must be used with any integer. The unit abbreviations can be uppercase or lowercase, and they can be specified in any order. Examples of valid *interval* values are

```
3d2H

3M2h

1d6h30m45s
```

To change the replication parameters defined for a fileset, use the options for the parameters you want to change. To change *all* replication parameters associated with a fileset, use the **-clear** option to remove all replication parameters previously defined for the fileset, and use the options for the parameters you want to change to indicate the new parameters. To have the system calculate default values for all replication parameters, use only the **-clear** option.

Use the **fts lsfldb** or **fts lsft** command to display the replication parameters associated with a read-write fileset. Use the **fts lsreplicas** command to display the statuses of replicas at replication sites. Use the **fts statrepserver** command to display the status of the Replication Server on a File Server machine.

Note that replication is available in a cell only if the following conditions have been met:

- **root.dfs**, the cell's main read-write fileset, is a DCE Local File System fileset
- **root.dfs** was mounted with an explicit read-write mount point as a subdirectory of the **root.dfs** fileset when the cell was configured
- **root.dfs** is replicated.

Refer to page 183 for information on configuring **root.dfs** to support replication.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset resides. The issuer DCE principal or DCE group must also be listed in the **admin.ft** file on the machine on which the read-write fileset resides if the following are true:

- The fileset's replication type is being changed from Release Replication to Scheduled Replication
- A replica actually resides at the replication site on the same File Server machine as the read-write fileset (the first replication site defined for a fileset that uses Release Replication must be on the same File Server machine as the read-write fileset).

Cautions

When using the **fts setrepinfo** command to set replication parameters, it is recommended that the default parameters (with the exception of MaxAge) be used for both types of replication. The dependencies between the parameters are complicated and should be defined by the issuer only when absolutely necessary.

Examples

The following command sets the initial Release Replication type and parameters for the read-write fileset named **applications.bin**. The default replication parameters are used for the fileset.

```
$ fts setrepinfo -fileset applications.bin -release
```

The following command changes the replication type for the **applications.bin** fileset from Release to Scheduled. It also clears the current replication parameters for the fileset and allows the system to calculate default values for all of the parameters.

```
$ fts setr -fileset applications.bin -scheduled -change -clear
```

The following command clears the current replication parameters used for the **applications.bin** fileset, replacing them with parameters specifed by the issuer of the command:

```
$ fts setr applications.bin -maxage 6h -failage 12h -reclaimwait 1d -minrepdelay 15m -clear
```

The previous command changes the default Scheduled Replication parameters as follows:

- It increases the MaxAge from the default of 2 hours to 6 hours.

- It decreases the FailAge from the default of the larger of 1 day or twice the MaxAge to 12 hours (twice the MaxAge).
- It increases the MinRepDelay from the default of 5 minutes or one- quarter of the DefaultSiteAge to 15 minutes.
- It increases the ReclaimWait from the default of 18 hours to 1 day.

Because the **-defaultsiteage** option is omitted from the command, the **-maxsiteage** option must be used when defining replication sites for the fileset with the **fts addsite** command.

Related Information

Commands:

fts addsite	**fts lsreplicas**	**fts statrepserver**
fts lsfldb	**fts release**	**fts update**
fts lsft		

fts statftserver

Purpose

Reports on the activity of a Fileset Server.

Format

fts statftserver -server *machine* **[-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Names the File Server machine about whose Fileset Server information is to be reported. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command.

  If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

  If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts statftserver** command reports on the actions of the Fileset Server (**ftserver** process) on the File Server machine specified with the **-server** option. The command returns information about the actions of the Fileset Server at the moment it is issued. This command is useful mainly if there is concern that a Fileset Server is not performing requested actions.

If no transactions are active on the specified machine, the command displays a message to that effect. This indicates that the Fileset Server is functioning properly. If transactions are active on the machine, the command displays information about the action currently being performed by the Fileset Server. Depending on the information displayed, the Fileset Server may or may not be functioning properly.

Output

If the Fileset Server is not currently performing any actions, the command displays the following message, indicating that the Fileset Server is functioning normally:

```
No active transactions on machine_name
```

If the Fileset Server is currently performing an action, the command displays information about the actions of the Fileset Server. The output includes fields containing ID numbers and flags that the Fileset Server sets for internal use. The details of the information returned by the command are more useful to programmers than to system administrators. A full understanding of the output requires familiarity with the code for the Fileset Server.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine specified by **-server**.

Related Information

Command:
ftserver

fts statrepserver

Purpose

Displays the status of a Replication Server.

Format

fts statrepserver -server *machine* **[-long] [-cell** *cellname*] **[{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
: Names the File Server machine about whose Replication Server status information is to be displayed. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-long
: Specifies that more detailed information about the Replication Server is to be displayed. The additional output includes information about each replica managed by the Replication Server on the specified machine.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command.

  If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other options specified with this option are ignored.

Usage

The **fts statrepserver** command displays information about the status of the Replication Server (**repserver** process) on the File Server machine specified with the **-server** option. Include the **-long** option to specify more detailed information about the Replication Server on the specified machine, as well as information about each replica managed by the Replication Server. Use the **fts lsreplicas** command to check the status of each replica of a fileset.

Privilege Required

No privileges are required.

Related Information

Commands:

fts lsreplicas **repserver**

fts syncfldb

Purpose

Synchronizes FLDB entries to match their fileset headers.

Format

fts syncfldb -server *machine* **[-aggregate** *name***] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
Names the File Server machine from which to compare filesets to entries in the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** for which to compare filesets to FLDB entries. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. Do not use this option under normal circumstances; omitting it allows synchronization of all filesets on **-server**. Use it only when just a single aggregate needs to be synchronized.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command.

If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts syncfldb** command inspects the fileset header of each online fileset that resides on a specified File Server machine (or, optionally, a specified aggregate or partition on that File Server machine). The command then checks that each FLDB entry is consistent with its fileset header. If the command encounters an inconsistency between a fileset header and its FLDB entry, the FLDB entry is changed to reflect the information in the fileset header. If the command encounters an FLDB entry without a corresponding fileset header, it deletes the FLDB entry; if the command encounters a fileset header without a corresponding FLDB entry, it creates an FLDB entry for that fileset.

The **fts syncfldb** command also performs the following additional functions:

- If it finds a backup fileset whose read-write source no longer exists at the same site, it displays a warning message.
- If it finds a fileset ID number that is larger than the value of the counter used by the Fileset Location Server (**flserver**) when allocating fileset ID numbers it records this ID number as the new value of the counter. The next fileset to be created receives a fileset ID number one greater than this number.
- If necessary, it increments or decrements the number of fileset entries recorded as residing on a File Server machine in the FLDB entry for the server.

The **fts syncfldb** command checks either all of the fileset headers on the File Server machine specified with the **-server** option or only the filesets on the optional partition or aggregate specified with the **-aggregate** option. The command checks a fileset header only if the fileset is marked as being `On-Line`. If the command encounters a busy fileset on an aggregate, it exits without checking any other filesets. (A busy fileset is one upon which a fileset-related operation such as a move, clone, or release is currently being performed.)

It is recommended that the **fts syncfldb** command be run on all File Server machines in a cell before the **fts syncserv** command is run on the File Server machines in the cell. However, nothing prohibits the commands from being executed in the reverse order or independently of each other.

Note that the **fts syncfldb** and **fts syncserv** commands cannot restore replication information lost when the entry for a DCE Local File System fileset is removed from the FLDB. Replication information must be reconstructed with the **fts setrepinfo** and **fts addsite** commands.

Because non-Local File System filesets do not have fileset headers, the **fts syncfldb** and **fts syncserv** commands have limited effectiveness on non-Local File System filesets. For example, because non-Local File System filesets do not have fileset headers, the **fts syncfldb** command cannot determine the name of a non-Local File System fileset that has no FLDB entry. If the command determines that it needs to create an FLDB entry for a non-Local File System fileset, it generates a name of the form **SYNCFLDB-ADDED-***number*, where *number* is a unique number appended to the name to differentiate it from other names of the same type. The **fts rename** command then needs to be used to rename the fileset to its original name.

Important Note to Users

In OS/390 DFS, if you issue **fts syncfldb** or **fts syncserv** command for a non-Local File System fileset, the unique name that gets generated is *not* of the form **SYNCFLDB-ADDED-***number*. Instead, it uses the actual name of the OS/390 HFS data set or RFS data set (prefix) in upper case, then the **fts** command may be issued to rename it to its original name (if it was different).

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on each machine that houses a version of any fileset stored at the specified site (**-server** and optionally **-aggregate**). The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset stored at the specified site.

Cautions

The physical disk on which a fileset resides cannot be moved from a machine in one cell to a machine in another cell with the expectation of simply running the **fts syncfldb** command to create an FLDB entry for the fileset in the new cell. Any attempt to introduce a fileset from one cell into another cell risks a fileset ID conflict between the newly introduced fileset and a fileset within the new cell that has the same fileset ID. This conflict causes one of the two conflicting filesets to be inaccessible.

Related Information

Commands:

fts addsite	**fts setrepinfo**	**fts syncserv**
fts rename		

Files:

devtab	**dfstab**

fts syncserv

Purpose

Synchronizes fileset headers to match their FLDB entries.

Format

fts syncserv -server *machine* **[-aggregate** *name***] [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-server *machine*
Names the File Server machine for which to check entries in the Fileset Location Database (FLDB). Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate or partition on **-server** for which to check FLDB entries. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. Do not use this option under normal circumstances; omitting it allows synchronization of all filesets on **-server**. Use it only when just a single aggregate needs to be synchronized.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command.

If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

If you use this option, do not use the **-noauth** option.

-verbose
Directs **fts** to provide detailed information about its actions as it executes the command.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

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 header marked as `Off-line`, but its FLDB entry lists it as being `valid`, the fileset is placed `On-line`.

The **fts syncserv** command checks either all of the filesets on the File Server machine specified with the **-server** option or only the filesets on the optionally specified partition or aggregate specified with the **-aggregate** option. The command also checks the reported sites of all copies of an inspected fileset (even though that requires checking filesets on server machines other than **-server**).

It is recommended that the **fts syncfldb** command be run on all File Server machines in a cell before the **fts syncserv** command is run on the File Server machines in the cell. However, nothing prohibits the commands from being executed in the reverse order or independently of each other.

Note that the **fts syncserv** and **fts syncfldb** commands cannot restore replication information lost when the entry for a DCE Local File System fileset is removed from the FLDB. Replication information must be reconstructed with the **fts setrepinfo** and **fts addsite** commands.

Because non-Local File System filesets do not have fileset headers, the **fts syncserv** and **fts syncfldb** commands have limited effectiveness on non-Local File System filesets. For example, because the **fts syncserv** command cannot destroy a disk partition, it cannot delete a non-Local File System fileset, even if it determines that the fileset needs to be deleted. Instead, the **fts** program displays an error message reporting the non-Local File System fileset that needs to be deleted to restore file system consistency. The proper commands then need to be used to delete the fileset.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on each machine that houses a version of any fileset stored at the specified site (**-server** and optionally **-aggregate**). The issuer's DCE principal or DCE group must also be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset stored at the specified site.

Examples

The following command synchronizes the FLDB entries of filesets whose site definitions mention **fs3**, including any copies of the filesets not located on **fs3**:

```
$ fts syncserv /.../abc.com/hosts/fs3
```

Related Information

Commands:

fts addsite	**fts setrepinfo**	**fts syncfldb**

File:
dfstab

fts unlock

Purpose

Unlocks an entry in the FLDB.

Format

fts unlock -fileset {*name* **|** *ID***} [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset *name* **|** *ID*
: Specifies the complete name or fileset ID number of the fileset whose entry in the Fileset Location Database (FLDB) is to be unlocked.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

  If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts unlock** command releases the lock on the FLDB entry for the fileset indicated by **-fileset**. Use the **fts unlockfldb** command to unlock multiple filesets at one time.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine on which a version of the fileset to be unlocked resides.

Cautions

Do not issue this command under normal circumstances. It is useful only if the FLDB entry for a fileset is locked but there is no reason to suspect inconsistency within the fileset or between it and the FLDB. Note that it is possible to list information from locked FLDB entries, even though they cannot be manipulated in other ways.

Examples

The following command unlocks the FLDB entry for the fileset named **user.terry**:

```
$ fts unlock user.terry
```

Related Information

Commands:

fts lock	**fts unlockfldb**

fts unlockfldb

Purpose

Unlocks all specified locked entries in the FLDB.

Format

fts unlockfldb [-server *machine***] [-aggregate** *name***] [-cell** *cellname***]**
[{-noauth | -localauth}] [-verbose] [-help]

Options

-server *machine*
: Names the File Server machine whose filesets are to have their Fileset Location Database (FLDB) entries unlocked. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option with **-aggregate** to unlock the entries for the filesets on a specific aggregate on **-server**. Omit both this option and **-aggregate** to unlock all of the entries in the FLDB.

-aggregate *name*
: Specifies the device name, aggregate name, or aggregate ID of an aggregate or partition on **-server** where the filesets whose FLDB entries are to be unlocked reside. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the **/opt/dcelocal/var/dfs/dfstab** file. The **-server** option must be specified with this option. Omit both this option and **-server** to unlock all of the entries in the FLDB.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth
: Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command.

  If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

  If you use this option, do not use the **-noauth** option.

-verbose
: Directs **fts** to provide detailed information about its actions as it executes the command.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts unlockfldb** command releases the locks on the FLDB entries indicated by the combination of options provided. To unlock:

- All entries in the FLDB, specify no options
- All entries that mention a File Server machine in a site definition, specify the name of the File Server machine with **-server**

- All entries that mention a specific site, specify both **-server** and **-aggregate**
- A single fileset, use the **fts unlock** command instead.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.fl** files on all Fileset Database machines or own the server entry for each machine that houses a version of any fileset to be unlocked.

Cautions

Do not issue this command under normal circumstances. It is useful only if FLDB entries for filesets at a certain site are locked, but there is no reason to suspect inconsistency within the filesets or between the filesets and the FLDB. Note that it is possible to list information from locked FLDB entries, even though they cannot be manipulated in other ways.

Examples

The following command unlocks all locked entries in the FLDB:

```
$ fts unlockfldb
```

Related Information

Commands:

fts lock	**fts unlock**

File:
dfstab

fts update

Purpose

Requests an immediate update of replicas of a read-write DCE Local File System fileset.

Format

fts update -fileset {*name* **|** *ID***} {-all | -server** *machine***} [-cell** *cellname***] [{-noauth | -localauth}] [-verbose] [-help]**

Options

-fileset {*name* **|** *ID***}**
: Specifies the complete name or fileset ID number of the read-write fileset whose replicas are to be updated immediately. For a fileset that uses Scheduled Replication, the command updates the indicated replicas to match the read-write version of the fileset. For a fileset that uses Release Replication, the command updates the replicas to match the read-only version that resides at the same site as the read-write version of the fileset. The fileset can use Release or Scheduled Replication.

-all Specifies that all replicas of **-fileset** are to be updated. Use this option or use **-server**.

-server *machine*
: Names a specific File Server machine on which the replica of the fileset indicated with the **-fileset** option is to be updated. Only the replica on the specified File Server machine is updated. Specify the file server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address. Use this option or use **-all**.

-cell *cellname*
: Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts update** command asks the Replication Server to make an immediate update of replicas of the read-write DCE Local File System fileset specified with the **-fileset** option. The effect of the command depends on whether the fileset to be updated uses Scheduled or Release Replication, as follows:

- *For a fileset that uses Scheduled Replication*, the command directs the Replication Servers on the indicated File Server machines to perform an immediate update based on the read-write version of the fileset. The Replication Servers ignore the value of the MinRepDelay parameter associated with the fileset; they immediately begin updating the replicas to match the version of the read-write fileset that exists at the time the command is issued.

- *For a fileset that uses Release Replication*, the command directs the Replication Servers on the indicated File Server machines to perform an immediate update based on the read-only replica that is stored on the same File Server machine as the read-write fileset (the replica that was created when the **fts release** command was last used for the fileset). The command does *not* first update the read-only replica on the read-write fileset's File Server machine. Because the MinRepDelay parameter does not apply to a fileset that uses Release Replication, the replicas should already be updated to match the replica on the read-write fileset's machine; the command should have no effect.

To indicate the replicas of the specified fileset that are to be updated use the command's **-all** or **-server** option as follows:

- To update all replicas of the specified fileset, use the **-all** option.
- To update the replica stored on a specific File Server machine, identity the machine with the **-server** option.

Note that, as with releasing a new version of a fileset that uses Release Replication, forcing an update of a fileset that uses Scheduled Replication does not ensure immediate access to data in the new version of the replica. A Cache Manager continues to provide data cached from the old version of the replica until the MaxAge for the fileset expires or until the Cache Manager needs to access data from the replica that it has not already cached.

To gain immediate access to data in the new version of the replica, issue the **cm flush** or **cm flushfileset** command to flush the old data from the cache. This forces the Cache Manager to replace data it has cached from the replica. However, because Replication Servers update replicas one at a time, the Cache Manager can still provide data from the old version of the replica if it accesses a replica that has yet to be updated. Until all replicas have been updated, you cannot directly force the Cache Manager to access data from the new version of the replica.

The **fts update** command does not change the replication type and parameters defined for the specified fileset. Before the **fts update** command can be used, the **fts setrepinfo** command must be used to define the replication parameters for the read-write fileset. The **fts addsite** command must also be used to define at least one replication site for the read-write fileset.

Use the **fts lsreplicas** command to check the status of replicas of the fileset. Use the **fts statrepserver** command to check the status of the Replication Server on a file server machine.

Privilege Required

No privileges are required.

Examples

The following command requests an immediate update of the replica of the read-write fileset named **applications.bin** at the replication site defined at the file server machine named **fs3**:

```
$ fts update applications.bin /.../abc.com/hosts/fs3
```

Related Information

Commands:

cm flush	**fts lsreplicas**	**fts setrepinfo**
cm flushfileset	**fts release**	**fts statrepserver**
fts addsite		

fts zap

Purpose

Removes a DCE Local File System fileset from its site without updating the FLDB.

Format

fts zap -ftid *ID* **-server** *machine* **-aggregate** *name* **[-cell** *cellname***] [{-noauth | -localauth}]**
[-verbose] [-help]

Options

-ftid *ID* Specifies the fileset ID number of the fileset to remove; a fileset name is not a valid argument.

-server *machine*
Names the File Server machine from which to remove the fileset. Specify the File Server machine using the machine's DCE pathname, the machine's host name, or the machine's IP address.

-aggregate *name*
Specifies the device name, aggregate name, or aggregate ID of the aggregate on **-server** from which to remove the fileset. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-cell *cellname*
Specifies the cell where the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **nobody** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
Directs **fts** to use the DFS server principal name of the machine on which the command is issued as the identity of the issuer. Use this option only if the command is issued from a DFS server machine (a machine that has a DFS server principal in the local Registry Database). You must be logged into the server machine as **root** for this option to work. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions as it executes the command.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **fts zap** command removes the DCE Local File System fileset with the fileset ID number specified by **-ftid** from the site defined by **-server** and **-aggregate**. It neither changes the corresponding Fileset Location Database (FLDB) entry for the fileset nor decrements the number of fileset entries recorded in the server entry in the FLDB for the specified File Server machine.

This command is useful in situations in which it is important to delete a fileset but, for some reason, the FLDB is unreachable (for example, the Fileset Location Server is unavailable). The issuer can remove information about the deleted fileset from the FLDB by running the **fts syncserv** and **fts syncfldb** commands. The issuer can also reconcile the FLDB with the **fts rmsite** command (which removes site information about a read-only version from a fileset's FLDB entry), the **fts delete** command (which removes site information about the read-write or backup version from a fileset's FLDB entry), or the **fts delfldbentry** command (which removes the entire entry for a fileset from the FLDB). (The **fts zap** command can also be used to remove normally temporary clone filesets that can sometimes be left after an interrupted **fts move** operation.)

If the DCE Local File System fileset to be removed is also mounted locally (as a file system on its File Server machine), you must remove its local mount point before you delete it. The **fts zap** command cannot be used to delete a fileset that is mounted locally.

Important Note to Users

On OS/390 DFS, you cannot mount a DCE Local File System fileset.

Privilege Required

The issuer's DCE principal or DCE group must be listed in the **admin.ft** file on the machine specified by **-server**.

Cautions

Do not use this command as the standard way to remove a fileset. It can make the FLDB inconsistent with the filesets on File Server machines. Use the **fts delete** command to remove the fileset entry from the FLDB at the same time that the fileset is removed.

Examples

The following command removes the fileset with fileset ID **0,,36870988** from **/dev/ufs1** on **fs6**, without recording the change in the FLDB:

```
$ fts zap 0,,36870988 /.../abc.com/hosts/fs6 /dev/ufs1
```

Implementation Specifics

You cannot mount a DCE Local File System fileset on OS/390 DFS.

Related Information

Commands:

fts delete	**fts rmsite**	**fts syncserv**
fts delfldbentry	**fts syncfldb**	

File:
dfstab

ftserver

Purpose

Initializes the Fileset Server.

Important Note to Users

On OS/390 DFS, the **ftserver** process is handled in an implementation-specific manner. The process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **ftserver**.

The **ftserver** command is not supported in OS/390 DFS.

For further information about adding or configuring the **ftserver**, refer to "bos create" on page 485.

Format

ftserver [-adminlist *filename*] **[-verbose] [-help]**

Options

-adminlist *filename*
Specifies the administrative list file that contains principals and groups authorized to execute **ftserver** RPCs (usually using **fts** commands). If this option is omitted, **ftserver** obtains the list of authorized users from the default administrative list file, **/opt/dcelocal/var/dfs/admin.ft**.

-verbose Directs the **ftserver** process to provide a detailed report on what it is doing by displaying messages on standard error.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **ftserver** command. See "bos help" on page 504 and "bos apropos" on page 484 for examples of using these commands.

Usage

The Fileset Server, or **ftserver** process, handles fileset administration operations, such as creating, deleting, moving, and replicating filesets. The **ftserver** process must be run on all machines that export data for use in the global namespace. A machine that runs the Fileset Server, the File Exporter (which is initialized by the **fxd** process), and **dfsbind** is considered a DFS File Server machine. The Fileset Server is started and controlled by the BOS Server.

The first time it is initialized, **ftserver** creates the **/opt/dcelocal/var/dfs/admin.ft** administrative list file if the file does not already exist. The principals and groups listed in the **admin.ft** administrative list are authorized to administer filesets on the machine. Because some operations, such as fileset moves, are accomplished by two Fileset Servers communicating, server principal names must also appear in the **admin.ft** list. For simplified administration, create one **admin.ft** administrative list that contains the server principal names of all machines in the administrative domain. The same **admin.ft** list can then be used by all **ftserver** processes in the domain.

When it is started, **ftserver** creates the **/opt/dcelocal/var/dfs/adm/FtLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when the **ftserver** is started, the process moves it to the **FtLog.old** file in the same directory (overwriting the current **FtLog.old** file if it exists) before creating a new version to append messages to.

Use the **fts statftserver** command to check the status of the Fileset Server on any server machine.

Privilege Required

You must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Output

If problems are encountered during initialization, the **ftserver** process displays error messages on standard error output. The **ftserver** process keeps an event log in the file **/opt/dcelocal/var/dfs/adm/FtLog**.

If run with the **-verbose** option, the **ftserver** process provides a detailed report on what it is doing by displaying messages on standard error.

Implementation Specifics

On OS/390 DFS, the **ftserver** process is handled in an implementation-specific manner. The process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **ftserver**.

The **ftserver** command is not supported in OS/390 DFS.

For further information about adding or configuring the **ftserver**, refer to “bos create” on page 485.

Related Information

Commands:

dfsbind	**fts statftserver**	**fxd**

Files:

admin.ft	**FtLog**

fxd

Purpose

Initializes the File Exporter and starts associated kernel daemons.

Important Note to Users

On OS/390 DFS, the **fxd** process is handled in an implementation-specific manner. The process runs as part of the **dfskern** process in the DFS server address space. **dfskern** is started and monitored by the **DFSCNTL** program in the DFS server address space by the OS/390 system commands: **START** and **MODIFY** (for further information, see Chapter 17, "OS/390 System Commands" on page 327).

The **fxd** command is not supported in OS/390 DFS.

The options described for **fxd** can be specified as parameters in the **ioepdcf** file (for further information, see "ioepdcf" on page 382).

Format

fxd -admingroup *group* **[-mainprocs** *number_of_background_daemons***]**
[-tokenprocs *number_of_token_daemons***] [-hostlife** *client_timeout***] [-hostrpc** *client_rpc_timeout***]**
[-pollinterval *server_poll_period***] [-maxlife** *max_hostlife***] [-maxrpc** *max_hostrpc***] [-notsr]**
[-minlocalprotectlevel *level***] [-maxlocalprotectlevel** *level***] [-minremoteprotectlevel** *level***]**
[-maxremoteprotectlevel *level***]**
[-verbose] [-help]

Options

-admingroup *group*
Specifies the name of the group that can administer the File Exporter on this machine. Members of the specified group can effectively change the permissions, owner, and owning group of any file system object exported from the machine. A group from the local cell can be specified by a full or abbreviated group name (for example, **/.../***cellname***/***group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name. The **-admingroup** option performs a function similar to that of the administrative lists associated with DFS server processes, such as the Fileset Server and the Fileset Location Server, that run in the user space.

-mainprocs *number_of_background_daemons*
Specifies the number of main kernel processes (File Exporter kernel daemons) to run on the machine. These File Exporter kernel daemons are responsible for receiving and servicing RPC requests for data and tokens from DFS clients. Specify an integer greater than 0 (zero) to indicate the number of main kernel daemons to perform these services. If this option is omitted, four main kernel daemons perform these services.

-tokenprocs *number_of_token_daemons*
Specifies the number of token-revocation kernel processes (File Exporter kernel daemons) to run on the machine. These File Exporter kernel daemons are responsible for responding to RPCs from DFS clients that are themselves responding to token revocation requests. Specify an integer greater than 0 (zero) to indicate the number of kernel daemons to perform these services. If this option is omitted, two kernel daemons perform these services.

-hostlife *client_timeout*
Specifies the host lifetime the File Exporter assigns to each client that contacts it. The host lifetime is the length of time for which the File Exporter considers a client to be alive. Each

client must contact the File Exporter within this amount of time to renew its host lifetime. As long as a client's host lifetime has not expired, the File Exporter cannot revoke its tokens without its permission.

By default, the File Exporter assigns each client a host lifetime of 2 minutes. Specify an integer to indicate a number of seconds to serve as the host lifetime. The host lifetime must be greater than 0 (zero) and less than or equal to the maximum host lifetime (specified with the **-maxlife** option) and the host RPC lifetime (specified with the **-hostrpc** option).

-hostrpc *client_rpc_timeout*
Specifies the host RPC lifetime the File Exporter assigns to each client that contacts it. The host RPC lifetime is the length of time for which the File Exporter guarantees to attempt an RPC to a client before it revokes its tokens. The File Exporter can revoke the tokens of any client whose host RPC lifetime has expired without contacting the client.

By default, the File Exporter assigns each client a host RPC lifetime of 2 minutes. Specify an integer to indicate a number of seconds to serve as the host RPC lifetime. The host RPC lifetime must be greater than or equal to the host lifetime (specified with the **-hostlife** option) and less than or equal to the maximum host RPC lifetime (specified with the **-maxrpc** option).

-pollinterval *server_poll_period*
Specifies the polling interval the File Exporter assigns to each client that contacts it. The polling interval is the frequency with which each client that has tokens from the File Exporter is to poll it in the event that it cannot be reached. Each client sends an RPC to the File Exporter with this frequency until it can again contact it.

By default, the File Exporter assigns each client a polling interval of 3 minutes. Specify an integer greater than 0 (zero) to indicate a number of seconds to serve as the polling interval.

-maxlife *max_hostlife*
Specifies the maximum host lifetime the File Exporter can grant a client. A client can request a host lifetime larger than that specified with the **-hostlife** option, but the File Exporter will not grant a host lifetime greater than the value specified with this option.

By default, the File Exporter uses a value of 2 minutes as the maximum host lifetime. Specify an integer to indicate a number of seconds to serve as the maximum host lifetime. The maximum host lifetime must be greater than or equal to the host lifetime (specified with the **-hostlife** option) and less than or equal to the maximum host RPC lifetime (specified with the **-maxrpc** option).

-maxrpc *max_hostrpc*
Specifies the maximum host RPC lifetime the File Exporter can grant a client. A client can ask for a host RPC lifetime larger than that specified with the **-hostrpc** option, but the File Exporter will not grant a host RPC lifetime greater than the value specified with this option.

By default, the File Exporter uses a value of 2 minutes as the maximum host RPC lifetime. Specify an integer to indicate a number of seconds to serve as the maximum host RPC lifetime. The maximum host RPC lifetime must be greater than or equal to the host RPC lifetime (specified with the **-hostrpc** option) and the maximum host lifetime (specified with the **-maxlife** option).

-notsr
Specifies that the File Exporter is to forego token state recovery when it is restarted. If this option is specified, the File Exporter accepts requests for new tokens as soon as it can again be contacted by clients. By default, it provides a brief token state recovery period during which it accepts requests only to re-establish tokens from clients that held them before it was restarted. (It bases the duration of its period of token state recovery on the greater of its **-pollinterval** or **-maxlife**, plus 20 seconds.)

This option is useful primarily for debugging purposes. Use it sparingly, as it prevents the File Exporter from maintaining consistent token state across file server machine restarts.

-minlocalprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the File Exporter and clients within the same cell. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Security" on page 728.

-maxlocalprotectlevel *level*
: Specifies the maximum acceptable DCE RPC authentication level for communications between the File Exporter and clients in the local cell. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Security" on page 728.

-minremoteprotectlevel *level*
: Specifies the minimum acceptable DCE RPC authentication level for communications between the File Exporter and clients in foreign cells. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Security" on page 728.

-maxremoteprotectlevel *level*
: Specifies the maximum acceptable DCE RPC authentication level for communications between the File Exporter and clients in foreign cells. The *level* is set either as an integer value between 0 and 7, the complete string defining the authentication level, or an abbreviation of that string. For a description of the various DCE RPC levels, see "Security" on page 728.

-verbose
: Directs **fxd** to produce more detailed information about its actions during initialization and as it creates kernel daemons.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

  The **help** and **apropos** commands available with all command suites are also available with **fxd**. See "bos help" on page 504 and "bos apropos" on page 484 for examples of using these commands.

Usage

The **fxd** command initializes the File Exporter on a File Server machine and starts all kernel daemons, such as those for garbage collection, required by the File Exporter. During initialization, it also passes the File Exporter such information as the name of the local cell, information about the local Fileset Database machines, and the identity of the group that can administer the File Exporter. The File Exporter uses this information to communicate with other processes, such as the Fileset Location Servers (**flserver**) on Fileset Database machines, and to ensure that only privileged users administer data in filesets exported from the machine.

The File Exporter must be run on all machines that export data for use in the global namespace. A machine that runs the File Exporter, the Fileset Server (**ftserver** process), and **dfsbind** is considered to be a DFS File Server machine.

The **-mainprocs** and **-tokenprocs** options can be used to alter the default number of kernel daemons running on the server machine, as follows:

- The **-mainprocs** option specifies the number of main kernel daemons that run on the machine to service RPC requests for data and tokens from DFS clients. The default number of main kernel daemons is four, which is usually sufficient to handle RPC requests from many DFS client machines. Use the **-mainprocs** option to increase the number of main kernel daemons dedicated to servicing RPC requests if the machine is to support a large number of DFS clients.

- The **-tokenprocs** option specifies the number of kernel daemons dedicated to responding to RPCs from DFS clients that are themselves responding to token revocation requests from the File Exporter. The default number of kernel daemons dedicated to this task is two. If the **-mainprocs** option is used to increase the number of main kernel daemons, use the **-tokenprocs** option to increase the number of kernel daemons dedicated to handling responses to token revocation requests accordingly.

On most system types, these kernel daemons appear as nameless entries in the output of the **ps** command (or its equivalent). However, because some of the kernel daemons run as threads rather than processes, not all of them show up in the output of the **ps** command.

The **-admingroup** option is used to associate system administrators with the **fxd** process. Members of the group specified with the **-admingroup** option have the necessary ACL and UNIX permissions to change the permissions of any file or directory object exported from the machine. They have the equivalent of the ACL **c** permission on the objects in each exported DCE Local File System fileset, and they can effectively change the mode bits on the objects in each exported non-Local File System fileset. (To change the permissions on an object that resides in a lower-level directory of an exported fileset, a member of the group may need to provide the group with the necessary permissions on directories in the path that leads to the object.) Members of the group can also change the owner and owning group of any object exported from the machine. Note that, while similar in many respects, inclusion in the group specified with the **-admingroup** option and being logged in as **root** are *not* equivalent. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Place only highly trusted users in the group associated with the **fxd** process. Members of the group generally constitute a subset of the users in other DFS administrative lists such as the **admin.bos** file. For simplified administration, the same group can be specified with the **-admingroup** options of all **fxd** commands issued in a domain.

The **fxd** command includes a number options that affect the File Exporter's management of tokens. The following two sections describe only those token-related issues germane to **fxd**'s options. Tokens, their management by the File Exporter, and their benefits and implications are described in “Data Access Management in DFS” on page 64. Refer to that section for more-detailed information about tokens and token management.

Token Management: Token management refers to the File Exporter's use of tokens to synchronize access to data and metadata on a File Server machine. The File Exporter uses tokens to track the clients that have accessed data from the machine and the types of operations they are permitted to perform on the data. When a client wants to access or change data on a File Server machine, it contacts the File Exporter on the machine to request the necessary tokens for the data. If the File Exporter can grant the client the requested tokens, the client in turn can use the tokens to access the data from the File Exporter.

Many factors affect the File Exporter's ability to grant a client's request for tokens. The File Exporter can always grant requests for tokens that do not conflict with those already held by another client. If requested tokens do conflict with existing tokens held by another client, the File Exporter tries to revoke the existing tokens. If it can revoke the existing tokens, it grants those requested; if it cannot, it either places the request in a queue or refuses it (the choice is the client's).

When its tokens are revoked, a client such as the Cache Manager flushes cached data for which the tokens applied, writing any modified data back to the File Server machine. Among the factors that affect the File Exporter's ability to revoke existing tokens are the various lifetimes it associates with the tokens it grants and the clients to which it grants them. The following list briefly introduces these values, of which the latter two can be modified with options of the **fxd** command:

Token lifetime
Specifies the length of time for which a token is valid. The File Exporter needs to revoke only valid tokens. Once a token has expired, the File Exporter does not need to revoke it; it can simply grant new tokens as if the expired token did not exist.

Host lifetime
Specifies the length of time for which the File Exporter considers a client to be alive. A client must contact the File Exporter to renew its host lifetime before it expires. As long as a client's host lifetime has not expired, the File Exporter cannot revoke its tokens without its permission.

Host RPC lifetime
Specifies the length of time for which the File Exporter agrees to attempt to make an RPC to a client before it revokes its tokens. The client's response to the RPC renews its host lifetime, meaning the File Exporter cannot revoke its tokens without its permission. If the client fails to respond to the RPC but its host lifetime has not expired, the File Exporter cannot revoke its tokens; if it fails to respond and its host lifetime has expired, the File Exporter can revoke any tokens it holds without contacting it further. The File Exporter can revoke a client's tokens without contacting it once its host RPC lifetime has expired. A client's host RPC lifetime must be at least as long as its host lifetime.

In general, the following rules apply to the File Exporter's revocation of valid tokens:

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

The following options of the **fxd** command can be used to modify the lifetimes the File Exporter assigns to its clients. By default, the File Exporter use values of 2 minutes for each of these lifetimes.

-hostlife Specifies each client's default host lifetime. The **-hostlife** must be greater than 0 (zero) and less than or equal to both the **-maxlife** and the **-hostrpc**.

-maxlife Specifies the maximum host lifetime the File Exporter will grant to a client that asks for one larger than the default specified with the **-hostlife** option. The **-maxlife** must be greater than or equal to the **-hostlife** and less than or equal to the **-maxrpc**.

-hostrpc Specifies each client's default host RPC lifetime. The **-hostrpc** must be greater than or equal to the **-hostlife** and less then or equal to the **-maxrpc**.

-maxrpc Specifies the maximum host RPC lifetime the File Exporter will grant to a client that asks for one larger than the default specified with the **-hostrpc** option. The **-maxrpc** must be greater than or equal to both the **-maxlife** and the **-hostrpc**.

If you use one of these options to modify a default lifetime value, be careful not to violate any of the dependency rules described in the previous list. In some cases, the command can adjust values not modified by the user to ensure that the dependencies are not violated, as follows:

- If you increase the value of **-hostlife** without specifying **-maxlife**, **-hostrpc**, or **-maxrpc**, the command increases the other three values as necessary.
- If you increase the value of **-maxlife** without specifying **-maxrpc**, the command increases the value of **-maxrpc** as necessary.

- If you increase the value of **-hostrpc** without specifying **-maxrpc**, the command increases the value of **-maxrpc** as necessary.
- If you decrease the value of **-maxlife** without specifying **-hostlife**, the command decreases the value of **-hostlife** as necessary.
- If you decrease the value of **-maxrpc** without specifying **-hostrpc**, the command decreases the value of **-hostrpc** as necessary.
- If you specify multiple values that explicitly violate one or more of the dependency rules, the command fails.
- If you specify a value that implicitly violates one or more of the dependency rules and the command cannot adjust other values to compensate for the violation, the command fails.

The command displays an appropriate message if it adjusts a value that was not specified or if it fails because specified values violate the previously defined rules.

Token State Recovery: Token state recovery refers to clients regaining their tokens following a network failure or File Server machine restart. In either of these situations, each client that cannot contact the File Exporter polls the File Exporter at regular intervals. When it can again reach the File Exporter, the client attempts to recover tokens it had before it lost contact. The frequency with which each client tries to contact the File Exporter in these cases is defined with the **-pollinterval** option of **fxd**; by default, each client polls the File Exporter every 3 minutes.

In the case of a network failure, a client may be unable to prevent its host lifetime from expiring before it can again contact the File Exporter. Once communication is restored, the client must either reclaim its tokens or request new ones, as necessary. The client may need to compete for its tokens with other clients to which the tokens were granted while it could not reach the File Exporter.

In the case of a File Exporter restart, the File Exporter loses all knowledge of tokens it granted. For a brief period after it restarts, it refuses all requests for new tokens from all clients. During this period, it accepts requests only to re-establish tokens from those clients that held them before it was restarted. The File Exporter gives those clients that held tokens before it was restarted the chance to recover their tokens without having to compete with other clients that could request the same tokens.

The File Exporter bases the length of its period of token state recovery after a restart on the **-maxlife** or the **-pollinterval**, whichever is greater (it adds 20 seconds to the value it chooses to compensate for its own initialization time). The larger of these two values ensures that each client that had tokens has an opportunity to contact the File Exporter before the File Exporter accepts requests for new tokens from all clients. (Within this time, each client will contact the File Exporter either to renew its host lifetime or to poll the File Exporter.)

If the File Exporter receives many requests to re-establish tokens just prior to the end of its token state recovery period, it dynamically extends the original length of the period. If many clients continue to contact it during the extension, the File Exporter continues to extend the period incrementally, to a maximum of twice its original length.

(Note that if a client is restarted for any reason, it loses all knowledge of the tokens it possessed prior to the restart; recovery of its tokens is not possible.)

Security: The **-minlocalprotectlevel**, **-maxlocalprotectlevel**, **-minremoteprotectlevel**, and **-maxremoteprotectlevel** options set the minimum and maximum RPC authentication bounds for communications between the File Exporter and clients. These bounds are used in negotiating an RPC authentication level for communications with clients. Two sets of bounds are maintained: a set that governs communications with clients within the same cell, and a second set that governs communications with clients in foreign cells.

In operation, the File Exporter and client (Cache Manager) interact to arrive at a mutually acceptable authentication level for communications. The negotiation starts with an RPC using the initial authentication level sent from the Cache Manager to the File Exporter. If the initial authentication level is outside the minimum or maximum bounds set through the **fxd** command, the File Exporter returns a response to the Cache Manager specifying that the authentication level is either too low or too high. The Cache Manager then decreases or increases its authentication level accordingly and retries the RPC. This process continues until the Cache Manager either adjusts its RPCs to an acceptable security level or the File Exporter requests a security level below the minimum set at the Cache Manager (causing the Cache Manager to refuse communications with the File Exporter). Once the Cache Manager and File Exporter have negotiated a security level, the Cache Manager stores this information so that it does not need to renegotiate this level for further communications with the File Exporter.

In addition, administrators can also set advisory bounds on a per-fileset basis. At present, these advisory levels serve only to bias the Cache Manager's selection of an initial authentication level. Advisory bounds are set through the **fts setprotectlevels** command and are stored in the FLDB record for that fileset.

Note that the use of this command does not preclude communications with File Servers running earlier versions of DFS.

The various authentication levels are set by specifying either an integer value between 0 and 7, a complete string specifying the authentication level, or an abbreviation of that string as the *level* argument for the various command options. The following lists the various authentication levels:

- **rpc_c_protect_level_default** or **default** or **0**: Use the DCE default authentication level.
- **rpc_c_protect_level_none** or **none** or **1**: Perform no authentication.
- **rpc_c_protect_level_connect** or **connect** or **2**: Authenticate only when the Cache Manager establishes a connection with the File Server.
- **rpc_c_protect_level_call** or **call** or **3**: Authenticate only at the beginning of each RPC received.
- **rpc_c_protect_level_pkt** or **pkt** or **4**: Ensure that all data received is from the expected host.
- **rpc_c_protect_level_pkt_integrity** or **pkt_integrity** or **5**: Authenticate and verify that none of the data transferred has been modified.
- **rpc_c_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.
- **rpc_c_protect_level_cdmf_priv** or **cdmf_priv** or **7**: Perform authentication as specified by all of the previous levels (except 6) and also encrypt each RPC argument value. This level provides a lower level of packet privacy than **rpc_c_protect_level_pkt_privacy**.

Note that there is a trade-off between selecting higher security and performance. The higher levels of security require more overhead and increase the response time in file operations with File Servers.

The default values of the File Exporter and Cache Manager are such that, if they are not changed, the File Exporter and Cache Manager will negotiate to the packet level for local cell and packet integrity level for non-local cell access. The default File Exporter values are as follows:

- The default minimum authentication level for communications with clients in the local cell is set to none.
- The default maximum authentication level for communications with clients in the local cell is set to packet privacy.
- The default minimum authentication level for communications with clients in foreign cells is set to none.
- The default maximum authentication level for communications with clients in foreign cells is set to packet privacy.

The default Cache Manager settings are as follows:

- The default initial authentication level for communications with File Exporters in the local cell is set to packet.
- The default minimum authentication level for communications with File Exporters in the local cell is set to none.
- The default initial authentication level for communications with File Exporters in foreign cells is set to packet integrity.
- The default minimum authentication level for communications with File Exporters in foreign cells is set to packet.

Given that both Cache Manager default initial authentication levels are set to packet and that this level is within the default bounds set at the File Exporter, the default authentication level is therefore packet. If you set the minimum bound at the File Exporter higher than packet integrity, any Cache Managers from a version of DFS previous to OSF DCE 1.2.2 (DCE 2.2 for AIX) will not be able to communicate with that File Exporter.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Cautions

If you restart the File Exporter with **fxd**'s **-notsr** option, the File Exporter does not enter token state recovery; clients do not have a protected opportunity to re-establish their tokens after the restart. Similarly, if you restart the File Exporter using different values for the command's lifetime or polling interval values, the File Exporter may not remain in token state recovery long enough to provide all clients an opportunity to re-establish their tokens after it is restarted (until they re-establish contact with the File Exporter, clients continue to use the previous lifetime and polling interval values, which may be too long if the File Exporter is directed to use shorter values when it is restarted).

Output

The command sends error messages to standard error output (**STDERR**) if problems are encountered during initialization. It also displays error messages if you specify values for its lifetime-related options that violate the dependencies mentioned in the section on Token Management. Finally, it displays warning messages if it adjusts one or more of its lifetime values to compensate for an option you specify.

Implementation Specifics

On OS/390 DFS, the **fxd** process is handled in an implementation-specific manner. The process runs as part of the **dfskern** process in the DFS server address space. **dfskern** is started and monitored by the **DFSCNTL** program in the DFS server address space by the OS/390 system commands: **START** and **MODIFY** (for further information, see Chapter 17, “OS/390 System Commands” on page 327).

The **fxd** command is not supported in OS/390 DFS.

The options described for the **fxd** command can be specified as parameters in the **ioepdcf** file (for further information, see “ioepdcf” on page 382).

Related Information

Commands:

dfsbind	**ftserver**	**modify**
fts setprotectlevels	**ioepdcf**	**start**

growaggr

Purpose

Increases the size of a DCE Local File System aggregate.

Format

growaggr -aggregate *name* **[-aggrsize** *blocks*] **[-noaction] [-help]**

Options

-aggregate *name*
: Specifies the device name or aggregate name of the DCE Local File System aggregate whose size is to be increased. These names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file. The specified aggregate does not need to be exported, nor does any fileset on the aggregate need to be mounted locally or in the global namespace.

-aggrsize *blocks*
: Specifies the total number of 1024-byte blocks to be available on the specified aggregate. The number of 1024-byte blocks specified with this option cannot exceed the total size of the disk partition on which the aggregate resides, and it must be at least three DCE Local File System blocks greater than the current size of the aggregate. (The number of bytes in a DCE Local File System block is defined on a per-aggregate basis with the **-blocksize** option of the **newaggr** command when an aggregate is created.)

  Include the **-noaction** option with this option to determine if the specified aggregate size is valid without changing the current size of the aggregate. Omit both this option and the **-noaction** option to increase the size of the aggregate to the total size of the disk partition on which it resides.

  The current maximum aggregate size for an OS/390 DCE Local File System aggregate is 4 GB (4194304 1K blocks).

-noaction
: Used without the **-aggrsize** option, this option directs the command to display the total number of 1024-byte blocks on the disk partition on which the specified aggregate resides. Used with the **-aggrsize** option, this option determines if the specified aggregate size is valid. The current size of the specified aggregate is not affected if this option is used.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

  The **help** and **apropos** commands available with all command suites are also available with the **growaggr** command. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The **growaggr** command is used to increase the size of an existing DCE Local File System aggregate. The aggregate whose size is to be increased is specified with the **-aggregate** option.

The **-aggrsize** option is used to specify the total size to make the aggregate. Specify the size as a number of 1024-byte blocks. The size specified with this option cannot exceed the total size of the disk partition on which the aggregate resides. The specified size also must be at least three DCE Local File System disk blocks greater than the current size of the aggregate. If it is not, the command displays the minimum size in 1024-byte blocks that can be specified. (The number of bytes in a DCE Local File

System block is defined on a per-aggregate basis with the **-blocksize** option of the **newaggr** command when an aggregate is initialized. It must be a power of two between 4096 and 65,536.)

If the **-noaction** option is included with the command, the present size of the aggregate is not affected. Combine the **-aggrsize** and **-noaction** options to achieve the following results:

- Specify only the **-aggrsize** option to increase the size of the aggregate to the specified size, as described previously.
- Specify only the **-noaction** option to determine the total number of 1024-byte blocks on the partition on which the aggregate resides.
- Specify both the **-aggrsize** and **-noaction** options to determine if the size specified with the **-aggrsize** option is valid (within the limits defined previously).
- Omit both the **-aggrsize** and **-noaction** options to increase the size of the aggregate to the total size of the disk partition on which it resides.

In operating systems that support logical volumes, the command is useful for increasing the size of an aggregate when the size of the logical volume on which the aggregate resides is increased. It can also be used to increase the size of an aggregate that was deliberately made smaller than the size of the partition or logical volume on which it resides.

The command does not affect any data or filesets that already reside on the aggregate to be grown.

Privilege Required

If the **-noaction** option is not included with the command, the issuer must have both the read and write permissions for the device (disk partition) on which the specified aggregate resides; if the **-noaction** option is included with the command, the issuer needs only the read permission for the device on which the aggregate resides. An issuer who is logged in as **root** on the machine on which the aggregate resides always has the necessary privilege to issue this command.

Note: On OS/390 DFS, if the **-noaction** option is not included with the command, the issuer **must** have read permission to the **/opt/dcelocal/var/dfs/devtab** file and RACF update authority for the linear data sets (LDSs) that make up the aggregate's logical volume. If the **-noaction** option is included with the **growaggr** command, the issuer needs read permission to the **/opt/dcelocal/var/dfs/devtab** file and read permission to the linear data sets.

Implementation Specifics

This command may be run from TSO/E, the OS/390 shell, or submitted as a batch job.

There are two methods that can be used to enlarge a DCE Local File System aggregate after it has been created on OS/390 DFS. The first method may be used if the DCE Local File System aggregate is smaller than the linear data set on which it resides. The second method enlarges the DCE Local File System aggregate by allowing you to add an additional linear data set. Use this method if the DCE Local File System aggregate and the linear data set are equal in size.

The following procedure describes the process for enlarging DCE Local File System aggregates on OS/390 DFS:

1. If you are enlarging a DCE Local File System aggregate by adding an additional linear data set, include the name of the data set to be added in the logical volume of the aggregate to be enlarged. The linear data sets comprising the logical volume for DCE Local File System aggregates are specified in sequential file entries in the **devtab** file for each logical volume. The sequential file entry must be added to the **devtab** file prior to issuing the **growaggr** command.

2. Ensure that the DFS server address space, **DFS**, is active on the OS/390 system.
3. Verify that the DCE Local File System aggregate to be enlarged is **not** already exported to DFS. If the aggregate is exported, the following example shows how to detach a DCE Local File System aggregate from DFS. In the example, **lfs1** is the name of an exported DCE Local File System aggregate.

   ```
   dfsexport lfs1 -detach
   ```

   The **dfsexport** command to detach a DCE Local File System aggregate from DFS can also be issued as a TSO/E command by issuing the following:

   ```
   dfsxport lfs1 -detach
   ```

4. Specify the **total** number of 1024-byte blocks to be available for the specified DCE Local File System aggregate.
5. Execute the **growaggr** command by submitting JCL. Execute **growaggr** by submitting PDS member **growaggr** in DFS.SIOESAMP. Step one will create a new linear data set. To add an additional linear data set, update the data set name and space parameters before continuing. If a new linear data set is not required, omit this step. Step two will execute the **growaggr** command.
6. Ensure **growaggr** is run as **UID = 0**. This **must** be done to access the **devtab** file.
7. Submit the **growaggr** JCL to enlarge the aggregate.
8. The following command can then be issued to export the enlarged aggregate where **lfs1** is the name of the aggregate:

   ```
   dfsexport lfs1
   ```

The following is an example of the **growaggr** JCL:

```
//jobcard  JOB ,'DFS GrowAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*-----------------------------------------------------------------
//*
//*  Allocate a VSAM linear data set for use as a POSIX device.
//*
//*-----------------------------------------------------------------
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=X
//SYSUDUMP DD     SYSOUT=X
//AMSDUMP  DD     SYSOUT=X
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=volser
//SYSIN    DD     *
  DEFINE CLUSTER (NAME(x.y.z) VOLUMES(volser) -
    LINEAR CYL(10 0) SHAREOPTIONS(2) )
/*
//*-----------------------------------------------------------------
//*
//*  Extend the LDS as an aggregate.
//*
//*
//*  NOTES:
//*        - Note that the "dev" *must* be lower case!
//*        - Note that the extra '/' is required because
//*          runtime parameters are separated from program
//*          paramaters by a '/'.
//*
//*-----------------------------------------------------------------
//GROWAGGR EXEC   PGM=IOEGRWAG,
```

```
// PARM=('ENVAR("_EUV_HOME=/opt/dfslocal/home/growaggr")//dev/lfs10 -agx
//            grsize 14400')
//*
//SYSPRINT  DD    SYSOUT=H
//STDOUT    DD    SYSOUT=H
//STDERR    DD    SYSOUT=H
//SYSUDUMP  DD    SYSOUT=H
//*
```

The following example describes the TSO/E command entry method. The TSO/E command, **growaggr**, is issued. The **-noaction** option directs the command to display the total number of 1024-byte blocks on the disk partition on which the exported DCE Local File System aggregate resides, **lfs1**. Note that a percent sign (%) precedes the command. This is necessary to allow TSO/E to differentiate the **growaggr** DFS REXX procedure used to support the TSO/E command entry method for this command from the DFS load module of the same name.

```
%growaggr -aggregate /dev/lfs010 -noaction

growaggr:  Extend device /dev/lfs010.
aggregate /dev/lfs010 holds 7200 (1024B) blocks, 7200 assigned
growaggr: Extend Successful !
```

The following example demonstrates the **growaggr** command as entered from OS/390 UNIX. Note that the command must be entered in lower case. The **-noaction** option directs the command to display the total number of 1024-byte blocks on the disk partition on which the exported DCE Local File System aggregate resides, **lfs10**. The **-aggregate** option specifies the device name or aggregate name of the DCE Local File System aggregate whose size is to be increased (**lfs10**).

```
$ growaggr -aggregate lfs10 -noaction

growaggr:  Extend device /dev/lfs010.
aggregate /dev/lfs010 holds 7200 (1024B) blocks, 7200 assigned
growaggr:  Extend Successful !
```

Related Information

Commands:

newaggr	**salvage**

Files:

devtab	**dfstab**

mapid

Purpose

Creates the identity mapping file that establishes a relationship between a user's DCE user ID and OS/390 user ID.

Format

mapid *input-file output-file*

Options

input-file	The HFS pathname of the Identity Mapping input file. This file is created and maintained by the system administrator. For further information, see “Identity Mapping Input File” on page 380.
output-file	The HFS pathname of the Identity Mapping output file.

Usage

The **mapid** command creates the Identity Mapping file by running the **mapid** on the Identity Mapping input file. The Identity Mapping file is created as an HFS file.

The **mapid** command may be run from TSO/E, the OS/390 shell, or in batch. The pathname specified in the *output-file* parameter must also be specified in the **_IOE_MVS_IDMAP environment** variable of the **dfskern** process.

Identity mappings may be updated in two ways on OS/390 DFS: through the use of the Identity Mapping Output file, or through the use of RACF identity mapping functions (see “Mapping DCE User IDs to OS/390 User IDs” on page 174). The method used on your system is determined during configuration by the **_IOE_MVS_IDMAP_SAF** environment variable value (see “Using the RACF Identity Mapping Function Method” on page 178).

Privilege Required

The issuer must be logged in to DCE as the cell administrator. The issuer must also be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**. If foreign users are being added, the issuer of this command must also have authority to read the **/.:/sec/principal** directory of the foreign cells.

Examples

In the following command, the **mapid** command is entered from OS/390 UNIX. The path for the input file is **/opt/dfslocal/home/dfskern/idmap.i**. The output path is specified as **/opt/dfslocal/home/dfskern/idmap.o**.

```
$ mapid /opt/dfslocal/home/dfskern/idmap.i /opt/dfslocal/home/dfskern/idmap.o
```

In the following command, the **mapid** command is entered from TSO/E. The path for the input file is **/opt/dfslocal/home/dfskern/idmap.i**. The output path is specified as **/opt/dfslocal/home/dfskern/idmap.o**.

```
MAPID "/opt/dfslocal/home/dfskern/idmap.i" "/opt/dfslocal/home/dfskern/idmap.o"
```

An example of the JCL used to run the **mapid** program follows.

```
//*jobcard JOB MSGLEVEL=1
//*
//*******************************************************************
//MAPID    EXEC PGM=IOEMAPID,REGION=0M,TIME=1440,
//      PARM=('/DD:INFILE DD:OUTFILE')
//*******************************************************************
//* Parameters
//*******************************************************************
//INFILE   DD PATH='/opt/dfslocal/home/dfskern/dceprincipalxxxx.table',
//            PATHMODE=(SIRWXU,SIRGRP,SIXGRP),
//            PATHOPTS=(ORDONLY),PATHDISP=(KEEP)
//OUTFILE  DD PATH='/opt/dfslocal/home/dfskern/dceuuidxxxxxxxxx.table',
//            PATHMODE=(SIRWXU,SIRGRP,SIXGRP),
//            PATHOPTS=(ORDWR,OCREAT),PATHDISP=(KEEP)
//****************************************************************
//SYSOUT   DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
```

Related Information

File:
Identity Mapping input file

newaggr

Purpose

Initializes a DCE Local File System aggregate.

Format

newaggr -aggregate *name* **-blocksize** *bytes* **-fragsize** *bytes* **[-initialempty** *blocks***]** **[-aggrsize** *blocks***]** **[-logsize** *blocks***]** **[-overwrite]** **[-verbose]** **[-noaction]** **[-help]**

Options

-aggregate *name*
: Specifies the device name or aggregate name of the disk partition to be initialized as a DCE Local File System aggregate. These names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-blocksize *bytes*
: Specifies the number of bytes to be available in DCE Local File System blocks on the aggregate (also referred to as the blocking factor). The value provided must be a power of two between 4096 and 65,536.

The number controls how disks are addressed in DCE Local File System. No file larger than 2^{31} blocks can be read or written. (Other considerations, chiefly I/O speed versus disk utilization, also constrain the maximum file size.)

Note: For the DFS client disk cache, a blocksize of 8192 is recommended for the disk cache aggregate on OS/390. For further information, see “Disk Cache Configuration” on page 257.

-fragsize *bytes*
: Specifies the number of bytes to be available in DCE Local File System fragments on the aggregate. The value provided must be a power of two between 1024 and the number of bytes specified with **-blocksize**.

The unit of storage allocation in DCE Local File System is the fragment, so this value controls the granularity of storage allocated to files. In other words, it affects the amount of space lost due to fragmentation.

-initialempty *blocks*
: Specifies the number of DCE Local File System blocks that DCE Local File System leaves empty at the beginning of the disk partition when it initializes the aggregate. The value provided must be an integer between 0 (zero) and 65,536 divided by the number of bytes specified with **-blocksize**. For example, if the value provided with **-blocksize** is 4096, the value provided with **-initialempty** cannot exceed 16 (65,536 divided by 4096).

The empty blocks reserved with this option are often used for a bootstrapping program. For this reason, the reserved blocks are often referred to as bootblocks.

If this option is omitted, one block is left empty at the beginning of the partition.

-aggrsize *blocks*
: Specifies the total number of DCE Local File System blocks to be available on the aggregate. Because this value cannot exceed the size of the disk partition, it can be used only to restrict the size of the aggregate. It must be large enough to accommodate at least the log and any blocks left empty at the beginning of the partition.

If this option is omitted, the default is the total number of DCE Local File System blocks on the disk partition being initialized as a DCE Local File System aggregate.

The current maximum aggregate size for an OS/390 DCE Local File System aggregate is 4 GB (4194304 1K blocks).

-logsize *blocks*
Specifies the number of DCE Local File System blocks to be reserved for the log on the aggregate. This value cannot exceed the number of DCE Local File System blocks used for **-aggrsize**, and it must contain at least enough blocks for the log to be initially created.

If this option is omitted, the default is 1% of the total number of DCE Local File System blocks on the aggregate (the number of DCE Local File System blocks used for **-aggrsize**).

-overwrite
Specifies that any existing file system found on the partition can be overwritten when the aggregate is initialized. If this option is specified, an existing file system on the disk partition is automatically overwritten; the issuer is not prompted for confirmation.

If this option is omitted and an existing file system is found on the partition, the command displays a message informing the issuer that the **-overwrite** option must be used to overwrite an existing file system. It then terminates with an exit code of at least 16 without overwriting the existing file system.

Note: On OS/390 DFS, the define cluster statements in the job control language (JCL) must be either commented out or deleted before using the **-overwrite** option.

-verbose Directs the command to provide more information on its actions as it executes. The information is displayed on standard output (**STDOUT**) unless it is directed elsewhere.

-noaction Directs the command to display information about what it would do without actually modifying the partition. Include the other options as you would to actually execute the command. The command displays the default values it would use for its options and informs the issuer if the disk partition already contains a file system.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **newaggr** command. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The **newaggr** command is used to initialize a partition on the local disk of a machine for use as an aggregate with DCE Local File System. The partition to be initialized as a DCE Local File System aggregate is specified with the **-aggregate** option. The **newaggr** command formats the specified partition by creating the metadata structure used by DCE Local File System for access control list (ACL) support, logging, and multiple fileset operations. It also creates temporary space on the disk used by the DCE Local File System log for faster restarts after system failures.

An aggregate is a collection of DCE Local File System disk blocks made up of the space available on the partition on which it resides. Each disk block on an aggregate has a fixed size specified with the **-blocksize** option. The **-blocksize** option specifies the number of bytes in each DCE Local File System block. The value specified with this option must be a power of two between 4096 (4 kilobytes) and 65,536 (64 kilobytes).

Each block can be further decomposed into fragments. Each fragment on an aggregate has a fixed size specified with the **-fragsize** option. The **-fragsize** option specifies the number of bytes in each fragment. The value specified with this option must be a power of two between 1024 (1 kilobyte) and the value specified with the **-blocksize** option.

DCE Local File System manages blocks and fragments as variable-length containers for the storage of user and system data. It manages filesets created on the aggregate as logically independent collections of data. Each fileset consists of a hierarchical collection of files residing entirely within a single aggregate. DCE Local File System obtains blocks for each fileset from a common allocation pool. As a result, filesets can share blocks (if the blocks are copy-on-write or if each fileset uses only a fragment of the block).

The **-initialempty** option can be used to reserve a number of empty blocks at the beginning of a partition. The empty blocks are referred to as bootblocks because they are often used for bootstrapping programs. The value provided with the **-initialempty** option must be an integer between 0 (zero) and 65,536 divided by the value specified with the **-blocksize** option. By default, one block is left empty.

The **-aggrsize** option can be used to restrict the number of DCE Local File System blocks in the aggregate. By default, all of the blocks available on the disk partition to be initialized are used in the aggregate. The value specified with the **-aggrsize** option cannot exceed the size of the partition being initialized. It must be large enough to accommodate at least the log and any blocks left empty at the beginning of the partition.

The **-logsize** option can be used to specify the number of DCE Local File System blocks to be reserved for the log on the aggregate. By default, 1% of the total number of DCE Local File System blocks on the aggregate is reserved for the log. The value specified with the **-logsize** option cannot exceed the number of DCE Local File System blocks used for the **-aggrsize** option, and it must specify at least enough blocks for the log to be initially created.

DCE Local File System also reserves a variable amount of disk space on the aggregate. By default, DCE Local File System reserves two megabytes of disk space on an aggregate. However, no less than 1% or no more than 10% of the total size of an aggregate is ever reserved; for example, only 1.5 megabytes are reserved on an aggregate whose total size is only 15 megabytes.

Reserved disk space is used for internal purposes. For example, the reserved space is used to avoid potential problems with routine administrative operations such as fileset moves and clones. The reserved space is not directly accessible to users and administrators. Use the **fts aggrinfo** command to display the total amount of disk space, including the amount of reserved disk space, on an aggregate.

If an existing file system on the disk partition being initialized is to be overwritten, include the **-overwrite** option with the command. The option instructs the command to overwrite any data found on the partition. To prevent an existing file system from being overwritten, omit the **-overwrite** option. If the command encounters an existing file system, it stops the initialization procedure without overwriting the existing file system and reports that it found a file system on the partition. It also instructs you to include the **-overwrite** option with the command to overwrite the resident file system.

Use the **-noaction** option to have the command report whether the partition already contains a file system or to display the values it calculates for the **-aggrsize** and **-logsize** options without actually overwriting a file system or initializing the partition. Specify all of the command's options as you would to actually execute the command, and include the **-noaction** option to display the results of the command without modifying the partition.

The **newaggr** command must be used to initialize a disk partition before the partition can contain DCE Local File System filesets. After the disk partition is initialized as a DCE Local File System aggregate with this command, an entry can be created for the aggregate in the **dfstab** file, and it can be exported to the DCE namespace with the **dfsexport** command. DCE Local File System filesets can then be created on it with the **fts create** command and mounted in the global namespace with the **fts crmount** command.

Because the **newaggr** command overwrites all data on the partition being initialized, the partition must not be mounted locally and it should not contain data when the command is run. If the **newaggr** command is issued with the **-overwrite** option to create a DCE Local File System aggregate on a disk partition that already contains a file system, the previous file system is destroyed. However, the command fails if it is run on an aggregate or partition that is currently exported to the DCE namespace, or if it is run on an aggregate that houses a locally mounted fileset. (If necessary, the **dfsexport** command can be used to detach an aggregate or partition from the namespace.)

In operating systems that support logical volumes, the command can be used to initialize a logical volume as a DCE Local File System aggregate. In such cases, all of the command's functionality described here with respect to a disk partition applies to the logical volume.

Cautions

Do not use the **newaggr** command to create non-Local File System aggregates. Do not use the command on a partition that contains data you want to retain; the command destroys all data on any partition it initializes. Finally, do not use the command on a currently exported aggregate or partition, or on an aggregate that houses a locally mounted fileset; the command fails in these cases.

Privilege Required

If the **-noaction** option is not included with the command, the issuer must have both the read and write permissions for the device (disk partition) to be initialized as a DCE Local File System aggregate; if the **-noaction** option is included with the command, the issuer needs only the read permission for the specified device.

Note: On OS/390 DFS, if the **-noaction** option is not included with the command, the issuer **must** have read permission to the **/opt/dcelocal/var/dfs/devtab** file and RACF update authority for the linear data sets that make up the aggregate's logical volume. If the **-noaction** option is included with the **newaggr** command, the issuer needs read permission to the **/opt/dcelocal/var/dfs/devtab** file and read permission to the linear data sets.

Implementation Specifics

This command may be run from TSO/E, the OS/390 shell, or submitted as a batch job.

On OS/390 DFS, the define cluster statements in the job control language (JCL) must be either commented out or deleted before using the **-overwrite** option.

On OS/390 DFS, DCE Local File System aggregates consist of one or more formatted linear data sets. The linear data sets for each aggregate are specified in the **/opt/dcelocal/var/dfs/devtab** file and make up that aggregate's logical volume. A logical volume consists of one or more linear data sets which are formatted to create a single DCE Local File System aggregate to be exported by **dfskern**. Logical volumes may be defined by editing the **/opt/dcelocal/var/dfs/devtab** file and adding the following entry:

```
define_lfs n
```

The entry assigns the minor device number, *n*, to the device you are defining. A minor device number can be any integer greater than zero. Each logical volume should have a unique minor device number. The minor device number becomes part of the aggregate's device name.

After defining the logical volume, you can specify the linear data sets you want to include in the logical volume. The following example shows a completed entry in the **/opt/dcelocal/var/dfs/devtab** file. A logical volume has been defined as **/dev/lfs1**, with 1 being the device's minor device number. The logical volume includes three linear data sets: **LDS00001**, **LDS00002**, and **LDS00003**. Lines beginning with an asterisk (*) are comments.

```
*Devtab - Example Entry for a logical volume
define_lfs 1
DFS.DCELFS.AGGR001.LDS00001
DFS.DCELFS.AGGR001.LDS00002
DFS.DCELFS.AGGR001.LDS00003
```

The following procedure describes the process for initializing DCE Local File System aggregates on OS/390 DFS:

1. Verify that the DCE Local File System aggregate to be formatted is **not** already exported to DFS. If the aggregate is exported, the following example shows how to detach a DCE Local File System aggregate from DFS. In the example, **lfs1** is the name of an exported DCE Local File System aggregate.

   ```
   dfsexport lfs1 -detach
   ```

   The **dfsexport** command to detach a DCE Local File System aggregate from DFS can also be issued as a TSO/E command by issuing the following:

   ```
   dfsxport lfs1 -detach
   ```

2. Ensure **newaggr** is run as **UID = 0**. This **must** be done to access the **devtab** file.
3. Execute the **newaggr** command by submitting JCL. Execute **newaggr** by submitting PDS member **newaggr** in *xxx*.SIOESAMP data set (where *xxx* is installation dependent). Step one will create a new linear data set. Update the data set name and space parameters before continuing. Step Two will execute the **newaggr** command.

The following is an example of the **newaggr** JCL:

```
//jobcard  JOB ,'DFS NewAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------------------------
//*
//*  Allocate a VSAM linear data set for use as a POSIX device.
//*
//*-------------------------------------------------------------------
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=volser
//SYSIN    DD     *
  DEFINE CLUSTER (NAME(a.b.c) VOLUMES(volser) -
    LINEAR CYL(250 0) SHAREOPTIONS(2) )
  DEFINE CLUSTER (NAME(d.e.f) VOLUMES(volser) -
    LINEAR CYL(250 0) SHAREOPTIONS(2) )
/*
//*-------------------------------------------------------------------
//*
//*  Format the LDS as an aggregate.
//*
//*  This version of newaggr also "loads" the VSAM linear data
//*  set(s)s if they have not been previously loaded.
//*
//*  NOTES:
//*        - Note that the "dev" *must* be lower case!
//*        - Note that the extra '/' is required because
//*          runtime parameters are separated from program
//*          parameters by a '/'.
//*
//*-------------------------------------------------------------------
//NEWAGGR  EXEC   PGM=IOENEWAG,REGION=0M,
// PARM=('ENVAR("_EUV_FTRACE=0")//dev/lfs1 8192 1024 -verbose')
//*
//SYSPRINT  DD    SYSOUT=H
//STDOUT    DD    SYSOUT=H
//STDERR    DD    SYSOUT=H
//SYSUDUMP  DD    SYSOUT=H
//CEEDUMP   DD    SYSOUT=H
//*
```

The following example describes the TSO/E command entry method. The TSO/E command, **newaggr**, is issued. The device, **/dev/lfs1**, is specified to have 8192 bytes available in DCE Local File System blocks on the aggregate. A fragment size of 1024 bytes is specified. The **-verbose** option provides information on the command as it executes. Note that a percent sign (%) precedes the command. This is necessary to allow TSO/E to differentiate the **newaggr** DFS REXX procedure used to support the TSO/E command entry method for this command from the DFS load module of the same name.

```
%newaggr -aggregate /dev/lfs1 8192 1024 -verbose

 edsk_CheckAsBSD:
 edsk_CheckAsBSD(''/dev/lfs1'') returns 0.
  FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
 edsk_CheckAsEpisode(''/dev/lfs1'') returns 0; SB@0.
  SectorSize 4096; NumBigChunks 113; minBlkSize 128;
  minBlkCount 57600;
  FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
 *** Using default initialempty value of 1.
 /dev/lfs1: 113 65536-byte sectors, giving 3 superblocks.
 *** Using default number of (8192-byte) blocks: 899
 *** Defaulting to 13 log blocks                 (maximum of 1 concurrent tran
sactions).
 Device /dev/lfs1, major 1, minor 1; total 900 8192-byte blocks.
 Block size 8192, frag size 1024, firstBlock 1, nBlocks 899
 Principal superblock at block 8, nLogBlocks 13.
 2 exceptions (additional superblock locations) at:
   except:1:: bigblk 7, block 56
   except:2:: bigblk 46, block 368
 /dev/lfs1: epig_CreateAggregate() returned OK.
 Done.  /dev/lfs1 is now an Episode aggregate.
  SectorSize 4096; TotalBlocks 899; BlockSize 8192;
  FragmentSize 1024; FirstBlock 1; NLogBlocks 13;
  NumBigChunks 113; minBlkSize 128; minBlkCount 57600;
  FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
```

Export the new DCE Local File System aggregate by issuing the following command where **lfs1** is the name of the aggregate:

```
$ dfsexport lfs1
```

The following example describes the OS/390 UNIX command entry method. The OS/390 UNIX command, **newaggr**, is issued. Note that in this method, all lower case letters are used. The device, **lfs10**, is specified to have 8192 bytes available in DCE Local File System blocks on the aggregate. A fragment size of 1024 bytes is specified. The **-verbose** option provides information on the command as it executes.

```
$ newaggr -aggregate lfs1 -blocksize 8192 -fragsize 1024 -verbose

 edsk_CheckAsBSD:
 edsk_CheckAsBSD(''/dev/lfs1'') returns 0.
  FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
 edsk_CheckAsEpisode(''/dev/lfs1'') returns 0; SB@0.
  SectorSize 4096; NumBigChunks 113; minBlkSize 128;
  minBlkCount 57600;
  FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
 *** Using default initialempty value of 1.
 /dev/lfs1: 113 65536-byte sectors, giving 3 superblocks.
 *** Using default number of (8192-byte) blocks: 899
 *** Defaulting to 13 log blocks                 (maximum of 1 concurrent tran
 sactions).
 Device /dev/lfs1, major 1, minor 1; total 900 8192-byte blocks.
 Block size 8192, frag size 1024, firstBlock 1, nBlocks 899
 Principal superblock at block 8, nLogBlocks 13.
 2 exceptions (additional superblock locations) at:
   except:1:: bigblk 7, block 56
   except:2:: bigblk 46, block 368
 /dev/lfs1: epig_CreateAggregate() returned OK.
 Done.  /dev/lfs1 is now an Episode aggregate.
  SectorSize 4096; TotalBlocks 899; BlockSize 8192;
  FragmentSize 1024; FirstBlock 1; NLogBlocks 13;
  NumBigChunks 113; minBlkSize 128; minBlkCount 57600;
  FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
```

Related Information

Commands:

dfsexport	**growaggr**	**salvage**
fts aggrinfo		

Files:

devtab	**dfstab**

repserver

Purpose

Initializes the Replication Server process.

Important Note to Users

On OS/390 DFS, the **repserver** process is handled in an implementation-specific manner. This process runs in the DFS server address space under the control of the of the **bosserver** process. The **bos create** command allows you to add and configure the **repserver**.

The **repserver** command is not supported in OS/390 DFS.

For further information about adding or configuring the **repserver**, refer to “bos create” on page 485.

Format

repserver [-mainprocs *number_of_background_daemons*] **[-tokenprocs** *number_of_token_daemons*] **[-help]**

Options

-mainprocs *number_of_background_daemons*
Specifies the number of background daemons to run on the machine. These daemons are responsible for the bulk of the effort required to maintain read-only replicas on the local machine, as well as for receiving and servicing RPC requests from DFS clients. If this option is omitted, four background daemons perform these services.

-tokenprocs *number_of_token_daemons*
Specifies the number of background daemons dedicated to servicing token revocation RPC requests from File Exporters. If this option is omitted, four background daemons service token revocation requests.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **repserver** command. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The Replication Server, or **repserver** process, in conjunction with the Cache Manager, tracks the currency of replicas and updates the versions of data being used at each replication site. The **repserver** process is used in Release and Scheduled Replication, and must run on any machine that stores read-only replicas of read-write filesets. For simplified administration, run the **repserver** process on all File Server machines.

The **repserver** process is started and controlled by the BOS Server.

The **-mainprocs** and **-tokenprocs** options can be used to alter the default number of background daemons running on the server machine, as follows:

-mainprocs
Specifies the number of background daemons that run on the machine to maintain read-only replicas housed on the local machine and to service RPC requests from DFS clients. The

default number of background daemons is four. Use the **-mainprocs** option to increase the number of background daemons if the machine houses a large number of replicas.

-tokenprocs
Specifies the number of background daemons dedicated to handling token revocation RPC requests from the File Exporters on File Server machines. The default number of background daemons dedicated to this task is four. If the **-mainprocs** option is used to increase the number of background daemons dedicated to maintaining replicas and servicing RPC requests from DFS clients, use the **-tokenprocs** option to increase the number of background daemons dedicated to servicing token revocation requests from File Exporters.

When it is started, the **repserver** creates the **/opt/dcelocal/var/dfs/adm/RepLog** event log file if the file does not already exist. It then appends messages to the file. If the file exists when the **repserver** is started, the process moves it to the **RepLog.old** file in the same directory (overwriting the current **RepLog.old** file if it exists) before creating a new version to which to append messages.

Use the **fts statrepserver** command to check the status of the Replication Server on any server machine. Use the **fts lsreplicas** command to check the status of fileset replicas.

Privilege Required

The issuer must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Output

If problems are encountered during initialization, the **repserver** sends error messages to standard error output (**STDERR**). The **repserver** keeps an event log in **/opt/dcelocal/var/dfs/adm/RepLog**.

Implementation Specifics

On OS/390 DFS, the **repserver** process is handled in an implementation-specific manner. This process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **repserver**.

The **repserver** command is not supported in OS/390 DFS.

For further information about adding or configuring the **repserver**, refer to “bos create” on page 485.

Related Information

Commands:

bos create	**fts lsreplicas**	**fts statrepserver**

File:
RepLog

salvage

Purpose

Uses the DFS Salvager to recover, verify, or salvage the structure of a DCE Local File System aggregate.

Format

salvage -aggregate *name* **[-recoveronly] [{-verifyonly | -salvageonly}] [-force] [-verbose] [-help]**

Options

-aggregate *name*
: Specifies the device name or aggregate name of the DCE Local File System aggregate to be verified, recovered, or salvaged. These names are specified in the first and second fields of the entry for the aggregate in the **/opt/dcelocal/var/dfs/dfstab** file.

-recoveronly
: Directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate. See the **Usage** section for information about using and combining the command's options.

-verifyonly
: Directs the Salvager to verify the specified aggregate. The Salvager examines the structure of the aggregate to determine if it contains any inconsistencies, reporting any that it finds. See the **Usage** section for information about using and combining the command's options.

-salvageonly
: Directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate. See the **Usage** section for information about using and combining the command's options.

-force
: Executes the Salvager in non-interactive mode. By default, the Salvager prompts for confirmation before proceeding in certain situations (for example, if it believes an aggregate on which it is run may be a non-Local File System partition). Use this option to direct the Salvager to proceed with all operations without asking whether it should continue. Use this option with care; the Salvager's changes can be unpredictable if this option is used with an invalid aggregate.

-verbose
: Directs the Salvager to produce detailed information about the aggregate as it executes. The information is useful primarily for debugging purposes. It is displayed on standard output (which can be redirected). Use this option alone or with any combination of the available options.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

  The **help** and **apropos** commands available with all command suites are also available with the **salvage** command. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The **salvage** command invokes the DFS Salvager on the DCE Local File System aggregate specified with the **-aggregate** option. Following a system restart, the Salvager employs the DCE Local File System log mechanism to return consistency to a file system by running recovery on the aggregate on which the file system resides. Recovery is the replaying of the log on the aggregate; the log records all changes made to metadata as a result of operations such as file creation and deletion. If problems are detected in the basic structure of the aggregate, if the log mechanism is damaged, or if the storage medium of the aggregate is suspect, the **salvage** command must be used to verify or repair the structure of the aggregate.

Use the command's **-recoveronly**, **-verifyonly**, and **-salvageonly** options to indicate the operations the Salvager is to perform on the specified aggregate, as follows:

Specify the **-recoveronly** option
: To run recovery on the aggregate without attempting to find or repair any inconsistencies found on it. Recovery is the replaying of the log on the aggregate. Use this option to quickly return consistency to an aggregate that does not need to be salvaged; this represents the normal production use of the Salvager. Unless the contents of the log or the physical structure of the aggregate is damaged, replaying the log is an effective guarantee of a file system's integrity.

Specify the **-verifyonly** option
: To determine whether the structure of the aggregate contains any inconsistencies without running recovery or attempting to repair any inconsistencies found on the aggregate. Use this option to assess the extent of the damage to an aggregate. The Salvager makes no modifications to an aggregate during verification. Note that it is normal for the Salvager to find errors when it verifies an aggregate that has not been recovered; the presence of an unrecovered log on an aggregate makes the findings of the Salvager, positive or negative, of dubious worth.

Specify the **-recoveronly** and **-verifyonly** options
: To run recovery on the aggregate and then analyze its structure without attempting to repair any inconsistencies found on it. Use these options if you believe replaying the log can return consistency to the aggregate, but you want to verify the consistency of the aggregate after recovery is run. Recovering an aggregate and then verifying its structure represents a cautious application of the Salvager.

Specify the **-salvageonly** option
: To attempt to repair any inconsistencies found in the structure of the aggregate without first running recovery on it. Use this option if you believe the log is damaged or replaying the log will not return consistency to the aggregate and may in fact further damage it. Under normal circumstances, do not salvage an aggregate without first recovering it.

Omit the **-recoveronly, -verifyonly,** and **-salvageonly** options
: To run recovery on the aggregate and then attempt to repair any inconsistencies found in the structure of the aggregate. Because recovery eliminates inconsistencies in an undamaged file system, an aggregate is typically recovered before it is salvaged. In general, it is good first to recover and then to salvage an aggregate if a machine panics or experiences a hardware failure.

: Omit these three options if you believe the log should be replayed before attempts are made to repair any inconsistencies found on the aggregate. (Omitting the three options is equivalent to specifying the **-recoveronly** and **-salvageonly** options.)

The following rule summarizes the interaction of the **-recoveronly**, **-verifyonly**, and **-salvageonly** options: The **salvage** command runs recovery on an aggregate and attempts to repair it unless one of the three options is specified; once one of these options is specified, you must explicitly request any operation you want the Salvager to perform on the aggregate.

The basic function of the Salvager is similar to that of the UNIX **fsck** program. The Salvager recovers a DCE Local File System aggregate and repairs problems it detects in the structure of the aggregate. It does not verify or repair the format of user data contained in files on the aggregate. If it makes changes, the Salvager displays the pathnames of the files affected by the modifications, when the pathnames can be determined. The owners of the files can then verify the files' contents, and the files can be restored from backups if necessary.

The Salvager verifies the structure of an aggregate by examining all of the anodes, directories, and other metadata in each fileset on the aggregate. An anode is an area on the disk that provides information used to locate data such as files, directories, ACLs, and other types of file system objects. Each fileset contains an arbitrary number of anodes, all of which must reside on the same aggregate. By following the links between the various types of anodes, the Salvager can determine whether the organization of an aggregate and the filesets it contains is correct and make repairs if necessary.

Not all aggregates can be salvaged. In cases of extensive damage to the structure of the metadata on an aggregate or damage to the physical disk that houses an aggregate, the Salvager cannot repair inconsistencies. Also, the Salvager cannot verify or repair damage to user data on an aggregate. The Salvager cannot detect problems that modified the contents of a file but did not damage the structure of an aggregate or change the metadata of the aggregate.

Like the UNIX **fsck** command, the Salvager analyzes the consistency of an aggregate by making successive passes through the aggregate. With each successive pass, the Salvager examines and extracts a different type of information from the blocks and anodes on the aggregate. Later passes of the Salvager use information found in earlier passes to help in the analysis.

Unlike the **fsck** command, the Salvager does not normally prompt the issuer for additional information as it executes. It typically performs the requested operation without prompting for input or pausing to verify any changes before it makes them. It prompts the issuer for confirmation only in the following cases:

- It believes the specified aggregate does not contain a DCE Local File System. This can occur if it finds a non-Local File System superblock whose creation time is more recent than the creation time of the DCE Local File System superblock.
- It finds that the size of the aggregate recorded in the DCE Local File System superblock exceeds the capacity of the partition on which the aggregate resides.

At the prompt, the issuer can choose to cancel or continue the operation. If the operation is continued under either of these circumstances and the aggregate proves to be invalid, unpredictable results can ensue. The best response in either case is to cancel the operation and attempt to determine the cause of the problem.

If you are confident that you want the salvager to continue in any case, you can include the **-force** option with the command. This option directs the Salvager to perform the requested operation without prompting the issuer for confirmation. Exercise caution when using the **-force** option, as the Salvager can produce unpredictable results if this option is used with an invalid aggregate.

In general, the Salvager exits with an error code of at least 16 without analyzing a partition that it is sure is not a DCE Local File System aggregate. It also exits with an error code of 16 if an aggregate to be recovered or salvaged is currently exported to the global namespace, or if a fileset on the aggregate to be recovered or salvaged is mounted locally. (If necessary, you can use the **dfsexport** command to detach an exported aggregate from the namespace.)

As the Salvager executes, it maintains a number of internal lists. Each list consists of anodes that failed verification in specific ways. When it initially scans an aggregate, the Salvager marks as "unsafe" anodes with which it encounters problems. The Salvager later attempts to determine the actual pathnames associated with these anodes to include the pathnames in the lists. When it has finished salvaging, the Salvager displays any non-empty lists. It also returns one of a number of informative exit codes, depending on the inconsistencies it found and the repairs it made. For more information about the lists and exit codes displayed by the Salvager see “Output.”

Internal structures maintained by the Salvager require a minimum of 1 megabyte of swap space. However, the total amount of swap space required by the Salvager depends largely on the size of the aggregate being salvaged and the extent of the damage to the aggregate.

Privilege Required

The privileges required depend on whether the **-recoveronly**, **-verifyonly**, or **-salvageonly** option is specified with the command: If just the **-verifyonly** option is included, the issuer needs only the read permission for the specified device (aggregate); if the **-recoveronly** or **-salvageonly** option is included, or if all three of these options are omitted, the issuer must have both the read and write permissions for the specified device.

Cautions

The Salvager can be used to salvage only DCE Local File System aggregates. If it is executed on a non-Local File System partition, it exits with an error code of at least 16 without performing any operations. Use the UNIX **fsck** program or its equivalent to verify or restore consistency to non-Local File System disk partitions.

Important Note to Users

The **fsck** program is not available in OS/390 UNIX.

By default, the Salvager asks for confirmation before proceeding with operations on aggregates that it suspects are non-Local File System partitions or whose indicated sizes exceed the capacities of the partitions on which they reside. The command's **-force** option can be used to direct the Salvager to continue without prompting in these cases. Do not include the **-force** option under normal conditions; the Salvager can make undesirable changes if the option is used with an invalid aggregate.

If the Salvager is used to recover or salvage an aggregate that is currently exported, it exits with an error code of 16 without performing the operation. Use the **dfsexport** command to detach an aggregate from the global namespace if necessary before recovering or salvaging it. (The Salvager can be used to verify the structure of a currently exported aggregate, but this is not a good practice; the results may be misleading.) The Salvager also exits with an error code of 16 if a fileset on an aggregate to be recovered or salvaged is mounted locally.

Output

The Salvager sends output to both **STDOUT** and **STDERR**. When it is started, it displays the device name of the aggregate on which it is run and the operation it is to perform. For example, the Salvager displays the following message if it is directed to recover an aggregate:

```
Will run recovery on <device>
```

Similarly, the Salvager displays the following message if it is directed to verify an aggregate:

```
Verifying <device>
```

If the **-verbose** option is specified with the command, the Salvager also generates the following information about the aggregate:

- Physical information about the configuration of the aggregate
- Header information from the aggregate, including the major and minor number of the device on which the aggregate was created, and the date and time at which the aggregate was created
- Information about how space in the aggregate is allocated, including:
  - The total size of the aggregate in blocks
  - The block size
  - The fragment size
  - The number of the first block in the aggregate
  - The location of the principal superblock for the aggregate
  - The number of logical blocks in the aggregate.

If the Salvager is used to recover an aggregate and the log on the aggregate does not need to be replayed, it displays only the introductor message described previously. If the log does need to be replayed and the Salvager can successfully recover the aggregate, it displays the following messages:

```
Recovery statistics
   <statistics>
Ran recovery on <device>
```

In the output, *statistics* consists of a few lines of information about the log and its replaying, and *device* is the device name of the aggregate. If it cannot run recovery for any reason, the Salvager displays an appropriate exit code (all Salvager exit codes are listed at the end of this section).

The Salvager can display much more output if it is asked to verify or salvage an aggregate on which it finds metadata errors. As it verifies or salvages a damaged aggregate, it displays a message similar to the following for each fileset in which it encounters metadata problems:

```
In volume <fileset> (avl <#integer>)
   in anode (<#integer>)
      <description>
```

It displays the first line once for each fileset, repeating the second and third lines once for each problem anode in the fileset. The output provides the following information:

fileset The name and ID number of each affected fileset.

avl *#integer*
A pointer to the anode for the fileset.

in anode *(#integer)*
A pointer to the anode for a file or other object in the fileset.

description
A brief description of the problem the Salvager found with the anode. If it was used to salvage the aggregate, the Salvager also describes any actions it took to repair the anode.

When it has finished executing, the Salvager lists each file whose metadata it found to be damaged, many of which it likely repaired if it salvaged the aggregate. For each file, it displays a line of the form:

```
<condition> <fileset:pathname> volume index:
<integer> anode index: <integer>
```

The output provides the following information:

condition A string that describes the state of the file or its metadata. (Information about the possible conditions follows this list.)

fileset The name of the fileset in which the affected file resides. In some cases, the Salvager cannot determine the fileset name.

pathname The pathname of the file, relative to the root directory of the fileset. In some cases, the Salvager cannot reconstruct the pathname for a file.

volume index
A pointer to the anode for the fileset. (This information can be used to identify earlier messages displayed by the Salvager that are related to this file.)

anode index
A pointer to the anode for the file. (This information can be used to identify earlier 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 such files contain all of their original data, these files should be restored from existing backups. This condition applies only to files on salvaged aggregates.

mayRestore
Files to which modifications were made (for example, files whose ACLs or property lists were changed). The owners of these files should verify their contents, or a system administrator should simply restore them from backups if a directory listing indicates that they have not been modified since the last backup was made. This condition also applies only to files on salvaged aggregates.

zeroLinkCnt
Files whose link counts should be 0 (zero). These files were deleted but not closed when the system crashed or were orphaned by the Salvager as it made repairs to the file system. The system will delete them when the aggregate is exported.

badLinkCnts
Files whose link counts were inconsistent with the number of references found to them. These files should be examined, if possible, or simply restored.

The Salvager can list a file more than once if it determines that multiple conditions apply to the file. It can also display one or more additional conditions (such as **badAcls** or **badPlists**), but files with which the additional conditions are associated are typically already covered by one or more of the conditions just described. Information in the additional lists is useful primarily for debugging purposes.

The Salvager also returns one of various exit codes to summarize its actions and findings. It returns the exit codes in the form of bits, which it uses to indicate the state of the aggregate. It can set multiple bits, but in general, the higher the bit, the greater the severity of the aggregate's problems (the higher bit always takes precedence when interpreting the output). The Salvager can return the following exit codes:

All bits off The Salvager found no problems. It displays a message that includes `Done` and `Checks out`. The command need not be run again.

First bit (**0x1**) set
The Salvager found one or more problems. It displays a message that includes `Done` and `Some inconsistencies found`. Run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Second bit (**0x2**) set
The Salvager found one or more problems and fixed them. It displays a message that includes `Done` and `Some inconsistencies repaired`. The command need not be run again. (Note that if the second bit is set, the first bit is usually also set; because the higher bit takes precedence, you do not need to run the command again.)

Third bit (**0x4**) set
: The Salvager found one or more problems and fixed some of them. It displays a message that includes `Incomplete` and `Some repairs made`. Some problems were more severe and require a subsequent salvage to be repaired; run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Fourth bit (**0x8**) set
: The Salvager found the aggregate to be irreparably damaged. It displays a message that begins `Problem`. Use the **newaggr** command to reinitialize the aggregate, and reconstruct the data from existing backups if possible.

Fifth bit (**0x10**) set
: The Salvager found some serious problem that prevents it from running on the aggregate; for example, the attempted recovery of the aggregate failed because of damage to the log, or the attempted salvage of the aggregate failed because the aggregate is not a DCE Local File System aggregate, it is currently exported, or it contains a locally mounted fileset. The Salvager displays a message that begins `Problem`. Attempt to determine the cause of the problem.

Including the **-verbose** option with the command produces more detailed information about the aggregate as the command executes. However, the additional information is useful primarily for debugging purposes.

Implementation Specifics

This command may be submitted from TSO/E, from the OS/390 shell, or as a batch job.

In the following example, the TSO/E command, **salvage** is issued. The **-verifyonly** option directs the DFS Salvager to verify the device, **/dev/lfs1**. The **-verbose** option directs the Salvager to produce detailed information as it executes. Note that a percent sign (%) precedes the command. This is necessary to allow TSO/E to differentiate the **salvage** DFS REXX procedure used to support the TSO/E command entry method for this command from the DFS load module of the same name.

```
%salvage -aggregate /dev/lfs1 -verifyonly -verbose

Verifying dev/lfs1
edsk_CheckAsBSD:
 SectorSize 4096; TotalBlocks 899; BlockSize 8192;
 FragmentSize 1024; FirstBlock 1; NLogBlocks 13;
 NumBigChunks 113; minBlkSize 128; minBlkCount 57600;
 FileSysCreateTime 830801643;
 FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
/dev/lfs1: Episode file sys created Mon Apr 29 18:14:03 1996
Device /dev/lfs1, major 1, minor 1; total 899 8192-byte blocks.
Block size 8192, frag size 1024, firstBlock 1, nBlocks 899
Principal superblock at byte 65536, nLogBlocks 13.
18:46:25.351682 (0099EA38 36) Starting work on device at 0x67c3a78.
18:46:25.375880 (0099EA38 36) This work will not salvage the aggregate as well.
18:46:25.438992 (0099EA38 36) AVL anode at 0x680e028
18:46:25.440487 (0099EA38 36) Aggregate created with newaggr
18:46:25.441321 (0099EA38 36) Disk format version is 1/3
18:46:25.537986 (0099EA38 36) Analyzing bitmap of 1 pages, 7248 bytes per page
, whose first bit describes physical block 0x1 on the
                (0099EA38 36) disk.
Processed 0 vols 4 anodes 0 dirs 0 files 0 acls
Done.  /dev/lfs1 checks out as Episode aggregate.
```

In the following example, the OS/390 UNIX command, **salvage** is issued. Note that the command is entered in all lower case letters for this method. The **-verifyonly** option directs the DFS Salvager to verify the device, **lfs1**. The **-verbose** option directs the Salvager to produce detailed information as it executes.

```
$ salvage -aggregate lfs1 -verifyonly -verbose

Verifying dev/lfs1
edsk_CheckAsBSD:
 SectorSize 4096; TotalBlocks 899; BlockSize 8192;
 FragmentSize 1024; FirstBlock 1; NLogBlocks 13;
 NumBigChunks 113; minBlkSize 128; minBlkCount 57600;
 FileSysCreateTime 830801643;
 FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''''
/dev/lfs1: Episode file sys created Mon Apr 29 18:14:03 1996
Device /dev/lfs1, major 1, minor 1; total 899 8192-byte blocks.
Block size 8192, frag size 1024, firstBlock 1, nBlocks 899
Principal superblock at byte 65536, nLogBlocks 13.
18:46:25.351682 (0099EA38 36) Starting work on device at 0x67c3a78.
18:46:25.375880 (0099EA38 36) This work will not salvage the aggregate as well.
18:46:25.438992 (0099EA38 36) AVL anode at 0x680e028
18:46:25.440487 (0099EA38 36) Aggregate created with newaggr
18:46:25.441321 (0099EA38 36) Disk format version is 1/3
18:46:25.537986 (0099EA38 36) Analyzing bitmap of 1 pages, 7248 bytes per page
, whose first bit describes physical block 0x1 on the
                (0099EA38 36) disk.
Processed 0 vols 4 anodes 0 dirs 0 files 0 acls
Done.  /dev/lfs1 checks out as Episode aggregate.
```

The following is an example of the **salvage** JCL:

```
//jobcard  JOB ,'LFS Salvager',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------------------------
//*
//*  Salvage an aggregate.
//*
//*  NOTES:
//*        - Note that the "/dev" *must* be lower case!
//*        - Note that the extra '/' is required because
//*          runtime parameters are separated from program
//*          parameters by a '/'.
//*        - Salvager must be run as root (OE UID 0)
//*        - Flag parameters are:
//*
//*           -recoveronly =  run log recovery, but don't salvage
//*           -verifyonly  =  don't modify contents of aggregate
//*           -salvageonly =  don't run log recovery, but do salvage
//*           -force       =  always salvage without confirmation
//*           -verbose     =  report more information during salvage
//*
//*-------------------------------------------------------------------
//SALVAGE  EXEC   PGM=IOESALVG,REGION=0M,
// PARM=('ENVAR("_EUV_HOME=/opt/dfslocal/home/salvage")//dev/lfs1 -vex
//             rifyonly -verbose')
//*
//SYSPRINT  DD    SYSOUT=H
//STDOUT    DD    SYSOUT=H
//STDERR    DD    SYSOUT=H
//SYSUDUMP  DD    SYSOUT=H
//*
```

Related Information

Commands:

dfsexport	**growaggr**	**newaggr**

Files:

devtab	**dfstab**

scout

Purpose

Initializes the **scout** program.

Format

scout -server *machine...* **[-basename** *common_prefix***] [-host] [-frequency** *seconds***]**
[-attention *stat/threshold_pair...* **] [-debug** *filename***] [-help]**

Options

-server *machine*
Names 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**).

-basename *common_prefix*
Specifies the DCE pathname prefix (for example, **/.../abc.com/hosts**) common to the File Server machines specified with the **-server** option. Do not include the / (slash) separating the prefix from the unique part of each machine name; it is included automatically with the **-basename** option. If a basename is specified with this option, it 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.

-host Displays the name of the machine running **scout** in the banner line. This is useful if you are logged into the machine remotely. By default, **scout** does not display this name.

-frequency *seconds*
Indicates how often the **scout** program is to probe the File Exporters. Specify a positive integer as a value in seconds; the default is 60 seconds.

Note: On OS/390 DFS, the **-frequency** option is ignored as **scout** probes the File Exporters only once.

-attention *stat/threshold_pair*
Specifies a list of attention settings (statistic and threshold value pairs). The **scout** program highlights any value for a statistic that exceeds its specified threshold; the highlighting is removed when the value goes below the threshold. The pairs can appear in any order. Legal statistic/threshold pairs are:

conn *connections*
The maximum number of connections that principals can have open to the the File Exporter before the value is highlighted. Enter a threshold for this statistic in the form of a positive integer.

fetch *number_of_fetches*
The maximum number of fetches (requests to send data) the File Exporter can service before the value is highlighted. Enter a threshold for this statistic in the form of a positive integer. The highlighting is removed when the File Exporter is restarted, at which time the value returns to 0 (zero). Enter a threshold for this statistic in the form of a positive integer.

store *number_of_stores*
The maximum number of stores (requests to store data) clients can send to the File Exporter for storage before the value is highlighted.

The number of stores is accumulated from the time the File Exporter is started. The highlighting is removed when the File Exporter is restarted, at which time the value returns to 0 (zero). Enter a threshold for this statistic in the form of a positive integer.

ws *active_client_machines*
The maximum number of active client machines the File Exporter can serve before the value is highlighted. Active indicates those machines that communicated with the File Exporter in the past 15 minutes. Enter a threshold for this statistic in the form of a positive integer.

disk *percent_full%*
The maximum percentage of an aggregate that can contain data before the value is highlighted. This threshold is applied to all exported aggregates and partitions on a File Server machine being monitored. Legal thresholds are the integers between 0 (zero) and 99; the default is 95%. *You must enter the % (percent sign) with this threshold.* If the % (percent sign) is absent, **scout** interprets the number as a number of kilobyte blocks. Use this threshold or use **disk** *minimum_blocks_free*.

disk *minimum_blocks_free*
The minimum number of kilobyte blocks that must be available on an aggregate before the value is highlighted. This threshold is applied to all exported aggregates and partitions on a File Server machine being monitored. Enter a threshold for this statistic in the form of a positive integer. Use this threshold or use **disk** *percent_full***%**.

Note: On OS/390 DFS, highlighting in **scout** is accomplished by appending a 'greater than' symbol, >, after the field to be highlighted.

-debug *filename*
Enables debugging output and directs it to the specified *filename*. Provide a complete pathname for *filename*. On OS/390 DFS, depending on the environment from which the **-debug** option is issued, the output may be directed to different locations. In the OS/390 UNIX environment, the current working directory is used by default. If this option is omitted, no debugging output is written.

Note: For detailed examples of the various ways **-debug** output can be directed on OS/390 DFS, see “Examples” on page 758.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with **scout**. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The **scout** command displays statistics gathered from the File Exporter running in the kernel on each File Server machine specified with the **-server** option. Usage statistics are also displayed about exported aggregates and partitions on the File Server machine being monitored. The **scout** program can be run on any DFS client or server machine. The binary file for the program resides in **/opt/dfsglobal/bin/scout**.

To change attention settings (statistic and threshold pairs), you must stop and restart the **scout** program. In addition, **scout** does not store the settings from previous executions; you must specify the desired settings each time you start the program.

Note: On OS/390 DFS, highlighting in **scout** is accomplished by appending a 'greater than' symbol, >, after the field to be highlighted.

Output for data is presented in line mode. Servers are probed once, output is displayed, and the program exits. If entered in the OS/390 shell, the program may be exited before completion by entering **Exit**.

On OS/390 DFS, debugging procedures vary slightly depending on the environment from which the user issues the **-debug** option. These environmental differences are explained in detail in “Examples.”

Output

The **scout** program on OS/390 DFS, displays information in line mode. The **scout** screen has three main parts:

- Banner Line
- Statistics Display Region
- Message/Probe Line.

The Banner Line at the top of the window or screen displays the word `Scout`, indicating the program is running. The name of the machine executing **scout** is displayed if the **-host** option is specified, and the basename of the File Server machines being monitored is displayed if the **-basename** option is specified.

The Statistics Display Region displays the statistics **scout** has gathered for each File Exporter. The region is divided into six columns, one column for each of the five statistic and threshold pairs used with the **-attention** option, and one column for the name of each File Server machine being monitored. In addition to highlighting any value that exceeds its specified attention threshold, **scout** highlights the name of any File Server machine whose File Exporter fails to respond to **scout**'s probes. The name remains highlighted until the machine resumes responding to **scout**'s probes.

The Message/Probe Line indicates how many times **scout** has probed the File Exporters for statistics. Use the **-frequency** option to specify how frequently **scout** is to probe the File Exporters.

Privilege Required

No privileges are required.

Examples

The following **scout** command causes the program to monitor the File Exporters on File Server machines **fs1** and **fs2** in the **abc.com** cell. The **scout** program probes the File Exporters every 30 seconds and writes debugging information to the file named **scout.one** in the current working directory.

```
$ scout -server fs1 fs2 -basename /.../abc.com/hosts -frequency 30 -debug scout.one
```

The following command causes **scout** to monitor the same two machines. The **scout** program highlights an entry in the `Fetch` column if the File Exporter services 20,000 or more fetches, and it highlights an entry in the `Store` column if the File Exporter accepts 10,000 or more stores.

```
$ scout -server fs1 fs2 -b /.../abc.com/hosts -attention fetch 20000 store 10000
```

Note: On OS/390 DFS, depending on the environment in which they are issued, there are differences in issuing the **-debug** option.

In the following OS/390 DFS example, the **scout** command is issued in the OS/390 shell environment. The **-server** option names the File Server machine to be monitored, **DCEDFS1**. The **-debug** option enables debugging and directs output to the specified file located in the user's current directory, **scout.debug**.

```
# scout -server DCEDFS1 -debug scout.debug
```

The following example is issued from a TSO/E environment. In this example, the output file, **scout.debug**, will be directed to the user's OS/390 UNIX home directory.

```
scout -server DCEDFS1 -debug scout.debug
```

In the TSO environments, debug output may be directed to an OS/390 or TSO/E data set. The following example shows the output file, **scout.debug**, redirected to a specified sequential OS/390 data set. The data set **must** be allocated in OS/390 **prior** to issuing this command. Note that the output file is preceded by double slashes, **//**. In this example, userid **SMITH** directs the debug output for server **DCEDFS1** to the OS/390 sequential data set named **SMITH.SCOUT.DEBUG**.

```
scout -server DCEDFS1 -debug //scout.debug
```

The following example shows output directed to a specifically named OS/390 Partitioned Data Set member, **OMVS.SCOUT(DEBUG)**. The data set **must** be allocated in OS/390 **prior** to issuing this command. Note that single quotes (**'**) surround the specified member name.

```
scout -server DCEDFS1 -debug //'OMVS.SCOUT(DEBUG)'
```

The following example directs the output to a file allocated to **ddname**. Debug information is written to the OS/390 sequential data set **OMVS.SCOUT.DEBUG**.

```
allocate file(scoutdd) da('omvs.scout.debug') shr reuse
scout -server DCEDFS1 -debug dd:scoutdd
```

The following is a sample of output from **scout** run from the TSO/E environment. In this example, the **scout** command is issued and the File Server machines, **DCEDFS1** and **dcecell2.endicott.ibm.com**, are to be monitored. The **-attention** option is specified by the **-att** entry. The settings chosen to be monitored are set to the **fetch**, **store**, **conn**, **disk**, and **ws** options. These options specify settings for the **scout** program to highlight. In this example, threshold values for these settings have been entered as **1**. Any values equal to or exceeding values of **1** in any of these settings for either of the File Exporters will be highlighted. The **-host** option causes **scout** to display the name of the host from where the command was issued in the banner line.

The lines immediately following the command string provide verification of command entries.

The output line containing the first banner line, **[DFS_HOST:] Scout**, identifies the host from which the command was issued, **DCEDFS1**. The categories selected to be monitored are shown. A connection is then made to the File Servers. The results are shown at the bottom. Values exceeding the specified thresholds are highlighted by the 'greater than' symbol, >, on the right side of the fields.

```
scout -server DCEDFS1 dcecell2.endicott.ibm.com -att fetch 1 store 1 conn 1 ws 1
 disk 1% -host
OES14541I (scout_AdoptThresholds) Setting fetch attention value to 1 (default -1)
 IOES14543I (scout_AdoptThresholds) Setting store attention value to 1 (default 1)
 IOES14535I (scout_AdoptThresholds) Setting connection attention value to 1 (default -1)
 IOES14545I (scout_AdoptThresholds) Setting workstation attention value to 1 (default -1)
 IOES14537I (scout_AdoptThresholds) New disk attention value: 0.1 used (default 0.950000)
 IOES14813I (fsprobe_Init) Host name from binding is dcefvt8.endicott.ibm.com
 IOES14813I (fsprobe_Init) Host name from binding is dcecellf1.endicott.ibm.com

 +------------------------------------------------------------------------------+
                               (DCEFVT8) Scout                                  >

 Conn     Fetch      Store    Ws      Server     Disk attn: > 1 pct. used
 ----    -------    -------  ----  ------------  ----------
 Conns Fetches    Stores     WrkStn  dcefvt8
 Conns Fetches    Stores     WrkStndcecellf1.e*


 Establishing File Server connection(s)...>

 +------------------------------------------------------------------------------+
                               (DCEFVT8) Scout                                  >

 Conn     Fetch      Store    Ws      Server     Disk attn: > 1 pct. used
 ----    -------    -------  ----  ------------  ----------
    1>         0          0     2>   dcefvt8     lfs8:19398
                                                 lfs12:86506                    >
                                                 ufs200:3040                    >
                                                 ufs210:59600                   >
                                                 ufs220:94036                   >
    1>       42>          0     2> dcecellf1.e*  lv04aggr:550                   >
                                                 njsaggr01:3239                 >
                                                 lv06aggr:2640                  >


 Probe 1 results>
 READY
```

Implementation Specifics

On OS/390 DFS, highlighting in **scout** is accomplished by appending a 'greater than' symbol, >, after the field to be highlighted.

On OS/390 DFS, output for data is presented in line mode. The **-frequency** option is ignored as **scout** probes the File Exporters only once, output is displayed, and the program exits. If entered in the OS/390 shell, the program may be exited before completion by entering **Exit**.

On OS/390 DFS, debugging procedures vary slightly depending on the environment from which the user issues the **-debug** option. These environmental differences are explained in detail in “Examples” on page 758.

udebug

Purpose

Displays Ubik status information relevant to the specified DFS database server.

Format

udebug -rpcgroup *RPC_server_group* **[-server** *machine***] [-long] [-help]**

Options

-rpcgroup *RPC_server_group*
: Specifies the RPC server group of the Ubik database servers whose status information you want to display. By convention, this is */.:/fs* for the **flserver** processes and */.:/subsys/dce/dfs/bak* for the **bakserver** processes.

-server *machine*
: Names the machine containing the database server whose Ubik status information is to be displayed; if a machine name is omitted, the command uses the name of the local machine. Specify the server machine using the full DCE pathname, abbreviated host name, or IP address.

-long Directs the command to provide additional information about the other database servers in the specified RPC server group. This flag is not necessary if the server specified with the **-server** option is the Ubik synchronization site because the information about the other database servers is provided automatically.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **udebug** command displays Ubik status information on the specified server in the specified RPC server group. If the specified server is the synchronization site or the **-long** flag is used with the command, the command displays information on all of the servers in the RPC server group.

Privilege Required

No privileges are required.

Output

The output for the **udebug** command always provides the following information for the specified database server:

- The IP address of the specified server machine. In the first example, this is `192.56.207.146`.
- The difference in seconds between the clock on the specified server machine and the machine on which the **udebug** command was run. In the first example, this is `0`.

  Note: If the message `****clock may be bad` appears, the difference between the two clocks is greater than 40 seconds, and you must synchronize the clocks on all of the server machines in the RPC server group. Refer to “Adding a Database Server Machine” on page 72 for more information on synchronizing the clocks.

- The IP address of the server machine that this server voted for to be the synchronization site and the time that the vote was cast. In the first example, this is `192.56.207.26` at `-10`.

  Note: Unless noted otherwise, all time is calculated and displayed as the number of seconds before (negative) or after (positive) the current time according to the clock on the local machine on which the **udebug** command is run.

- The time at which the last round of sync-site voting began. In the first example, this is `-11`.
- The version of the database in use on this server machine. In the first example, this is `750478963.1`.
- Whether the server is the synchronization site; if it is, the duration of the synchronization site status and the number of servers in the RPC server group are also provided. In the first example, the message `I am not sync site` indicates that the server is not the sync site.
- If the server is *not* the synchronization site, the following information is displayed:
  - The IP address of the lowest server in the RPC server group and the time that a beacon was last sent from that server to the specified server. In the first example, this is `192.56.207.26` at `-10`.
  - The IP address of the synchronization site and the time that a beacon was last sent from that server. In the first example, this is `192.56.207.26` at `-10`.

  If the server is the synchronization site, the current state of the server is displayed, using one of the following flags. In the second example, this is `1f`.
  - 1 - Indicates that the server is the synchronization site.
  - 3 - Indicates that the server is the synchronization site and that it has found the latest version of the database.
  - 7 - Indicates that the server is the synchronization site and that it has fetched the latest version of the database.
  - f - Indicates that the server is the synchronization site and a quorum has been reached in the RPC server group, but the synchronization site has not distributed the latest version of the database to all servers in the RPC server group.
  - 1f - Indicates that server is the synchronization site, a quorum has been reached in the RPC server group, and the synchronization site has distributed the latest version of the database to all servers in the RPC server group.
- The version of the database in use at the synchronization site. In the first example, this is `750478963.1`.
- The total number of database pages locked and the number of database pages locked for write purposes on the server. (Anything other than a 0 indicates database activity.) In the first example, this is `0` and `0`.
- The time that the server was the synchronization site, if it ever has been, or a message indicating that the server has never been the synchronization site. In the first example, the message `This server has never been sync site` indicates that the server has never been the synchronization site.

If the **udebug** command specifies the synchronization site of the RPC server group or if the **-long** option is used with the command, the following additional information is displayed for each of the other database servers in the RPC server group:

- The IP address of each server machine. In the second example, the first server machine listed has the IP address `192.56.207.36`.
- The version of the database in use on each server machine. (A 0.0 indicates that the server does not have a version of the database.) In the second example, the first server listed uses the database version `750478963.1`.

- The last time a vote was received from this server by the server specified with the **-server** option. In the second example, the server with IP address `192.56.207.26` received a vote from the first server with IP address `192.56.207.36` at `-8`.
- The last time a beacon requesting a vote was sent to each server. In the second example, the first server received a beacon at `-9`.
- The last vote, yes or no, cast by each server. In the second example, the first server cast a `yes` vote.
- A flag (`dbcurrent`) indicating whether the version of the database in use on each server machine is current with the synchronization site; 0 indicates no, 1 indicates yes. In the second example, the first server has a current version of the database.
- A flag (`up`) indicating whether the corresponding server process on each server machine is up; 0 indicates no, 1 indicates yes. In the second example, the first server is up.
- A flag (`beaconSince`) indicating whether a response (vote) to the latest beacon was sent by each server to the synchronization site. In the second example, the first server sent a response to the latest beacon.

Examples

The following command displays information on a specified database server that is not a synchronization site:

```
$ udebug /.:/fs fs2
```

```
Host 192.56.207.146, his time is 0
Vote: Last yes vote for 192.56.207.26 at -10 (sync site); Last vote started
at -11
Local db version is 750478963.1
I am not sync site
Lowest host 192.56.207.26 at -10
Sync host 192.56.207.26 at -10
Sync site's db version is 750478963.1
0 locked pages, 0 of them for write
This server has never been sync site
```

The following command displays information on a specified database server that is a synchronization site; the output also provides information on the other database servers in the RPC server group:

```
$ udebug /.:/fs fs4
```

```
Host 192.56.207.26, his time is 0
Vote: Last yes vote for 192.56.207.26 at -9 (sync site); Last vote started
at -9
Local db version is 750478963.1
I am sync site until 81 (4 servers)
Recovery state 1f
Sync site's db version is 750478963.1
0 locked pages, 0 of them for write
This server last became sync site at -38195

Server 192.56.207.36: (db 750478963.1)
   last vote rcvd at -8, last beacon sent at -9, last vote was yes
   dbcurrent=1, up=1 beaconSince=1

Server 192.56.207.146: (db 750478963.1)
   last vote rcvd at -8, last beacon sent at -9, last vote was yes
   dbcurrent=1, up=1 beaconSince=1
```

```
Server 192.56.207.94: (db 750478963.1)
   last vote rcvd at -8, last beacon sent at -9, last vote was yes
   dbcurrent=1, up=1 beaconSince=1
```

Related Information

Commands:

bakserver	**flserver**

upclient

Purpose

Initializes the client portion of the Update Server.

Important Note to Users

On OS/390 DFS, the **upclient** process is handled in an implementation-specific manner. This process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **upclient**.

The **upclient** command is not supported on OS/390 DFS.

For further information about adding or configuring the **upclient**, refer to “bos create” on page 485.

Format

upclient -server *machine* **-path {***filename* **|** *directory_name***}... [-time** *frequency***] [-file** *log_file***] [-verbose] [-help]**

Options

-server *machine*
: Specifies the DCE pathname of the machine (for example, **/.../abc.com/hosts/fs1**) whose files are to be periodically checked. The machine should be either the System Control machine for the cell or domain.

-path {*filename* **|** *directory_name***}**
: Names each file or directory to be checked periodically on the local disk of the machine with the **-server** option. If multiple paths are supplied, they must be unique, disjoint trees in the file system. Paths are examined from left to right. Paths that intersect with previous paths used in the command are logged as errors (if a log file is specified with **-log**) and ignored.

  If you specify a directory, the **upclient** process recusively checks all files and directories located beneath the specified directory. Therefore, you can specify a / (slash) with this option to check all files and directories on the local disk of the machine specified with the **-server** option.

Important Note to Users

When the **bos create** command is issued on OS/390 DFS, requests for different files from the same directory may be combined using the **-path** option. The trailing slash (/) after the pathname must be included before the file name. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The pathname and file names must be enclosed in parentheses (for example, **-path (/opt/dcelocal/var/dfs/ admin.ft admin.fl)**).

-time *frequency*
: Specifies in number of seconds how often the **upclient** process is to check each file or directory specified with **-path** for changes. The default is 300 seconds (5 minutes).

-file *logfile*
: Names the log file on the local machine to which errors are to be written. Because multiple **upclient** processes can be run on one machine, choose a distinct file name for the log. If this option is omitted, no errors are logged.

-verbose	Directs the **upclient** process to produce detailed information about its actions each time it checks for new versions of files (as specified with the **-time** option). The process lists each file and directory object it checks and any changes it makes to local versions of these objects. The output is sent to standard error.
-help	Prints the online help for this command. All other options specified with this option are ignored. The **help** and **apropos** commands available with all command suites are also available with the **upclient** command. See “bos help” on page 504 and “bos apropos” on page 484 for examples of using these commands.

Usage

The **upclient** command initializes the client portion (**upclient** process) of the Update Server. The **upclient** process periodically checks specified files and directories on the local disk of the server machine specified with the **-server** option to be sure they match the corresponding files and directories on the local machine (the machine running the **upclient** process). If a file on the specified server machine does not match the version on the local machine, the **upclient** process requests the newer version from the server portion (**upserver** process) of the Update Server on the specified machine. It then overwrites the local version of the file with the newer version.

The **-time** option specifies how often the **upclient** process is to check for changed versions of files and directories. The **-path** option specifies the files and directories the **upclient** process is to check. If you specify a directory, the **upclient** process recursively checks all files and directories located beneath the specified directory. To check all files and directories on the indicated server machine, specify a **/** (slash) with the **-path** option.

When the **bos create** command is issued on OS/390 DFS, requests for different files from the same directory may be combined using the **-path** option. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The pathname and file names must be enclosed in parentheses (for example, **-path (/opt/dcelocal/var/dfs/ admin.ft admin.fl)**).

If you specify multiple files and directories with the **-path** option, the paths must be disjoint (nonintersecting). Pathnames specified with the **-path** option are examined from left to right. Any path that intersects with a previous path is logged as an error (if a log file is named with the **-file** option) and ignored. An error also occurs if the **-path** option names a file or directory that the **upserver** process on the specified server machine is not directed to distribute. Because multiple **upclient** processes can be run on a single machine, a file name specified with the **-file** option must be distinct.

Finally, the machine running the **upclient** process must be named in the **admin.up** file on the machine running the **upserver** process (the machine specified with the **-server** option). Otherwise, its **upclient** process is not permitted to access files from the **upserver** process.

Privilege Required

You must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Implementation Specifics

On OS/390 DFS, the **upclient** command runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **upclient**.

The **upclient** command is not supported on OS/390 DFS.

For further information about adding or configuring the **upclient**, refer to “bos create” on page 485.

When the **bos create** command is issued on OS/390 DFS, requests for different files from the same directory may be combined using the **-path** option. The trailing slash (/) after the pathname must be included before the file name. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The pathname and file names must be enclosed in parentheses

In the following example, parameters are entered identifying the File Server machine **wichita** and the files the client is to check. The **-path** option specifies the files to be checked. Note that the full pathname, **(/opt/dcelocal/var/dfs/ admin.up admin.bos admin.ft)** is entered and that the pathname is surrounded by parentheses. In this example, the **admin.up**, **admin.bos**, and **admin.ft** files are specified and separated by spaces. All reside in the same directory. These are files containing administrative lists for the Update Server (**upserver**), the Basic OverSeer Server (**bosserver**) and Fileset Server (**ftserver**) processes. The **-f filename** parameter invokes a switch, **-f**, that points to a file called **UpCLog** that is to be used as a log file.

```
$ bos create /.:/hosts/DFSMVS upclient simple "upclient \
    envar('_EUV_HOME=/opt/dfslocal/home/upclient')/ >dd:upclient 2>&1 \
    -server /.:/hosts/wichita -path (/opt/dcelocal/var/dfs/ admin.up admin.bos admin.ft) \
    -f UpCLog"
```

Related Information

Commands:

bos create	**upserver**

File:
admin.up

upserver

Purpose

Initializes the server portion of the Update Server.

Important Note to Users

On OS/390 DFS, the **upserver** process is handled in an implementation-specific manner. This process runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **upserver**.

The **upserver** command is not supported on OS/390 DFS.

For further information about adding or configuring the **upserver**, refer to "bos create" on page 485.

Format

upserver -path {*filename* **|** *directory_name***}... [-adminlist** *filename***] [-help]**

Options

-path {*filename* **|** *directory_name***}**
Names each file or directory to be distributed (exported) in unencrypted form upon request. If multiple paths are supplied, they must be unique, disjoint trees in the file system. Paths are examined from left to right; paths that intersect with previous paths used in the command are logged as errors and ignored.

All files and subdirectories located beneath a specified directory can be distributed from the local machine. Therefore, you can specify a / (slash) with this option to allow all files and directories on the local disk of the machine to be distributed.

Important Note to Users

When the **bos create** command is issued on OS/390 DFS, requests for different files from the same directory may be combined using the **-path** option. The trailing slash (/) after the pathname must be included before the file name. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The pathname and file names must be enclosed in parentheses (for example, **-path (/opt/dcelocal/var/dfs/ admin.ft admin.fl)**).

-adminlist *filename*
Specifies the file that contains server principals authorized to request files from the local machine. If you do not specify the complete pathname of a file, the file is assumed to reside in the current working directory. If this option is omitted, the **upserver** uses the default file (**/opt/dcelocal/var/dfs/admin.up**).

-help
Prints the online help for this command. All other options specified with this option are ignored.

The **help** and **apropos** commands available with all command suites are also available with the **upserver** command. See "bos help" on page 504 and "bos apropos" on page 484 for examples of using these commands.

Usage

The **upserver** command initializes the server portion (**upserver** process) of the Update Server. The **upserver** process distributes files from the local disk of a machine in response to requests from the client portion (**upclient** process) of the Update Server running on other machines. An **upserver** process should be run on the System Control machine for the cell or domain.

The **-path** option specifies which of the files and directories on a machine's local disk the **upserver** can distribute. The **-path** option specifies the files and directories the **upserver** process can distribute from the local disk of the machine on which it is run. The **upserver** process can distribute all files and subdirectories located beneath a specified directory on the local machine; an **upclient** process can request and receive any file from the specified directory. Specify a **/** (slash) to allow all files and directories on the local disk of the machine to be distributed.

When the **bos create** command is issued on OS/390 DFS, requests for different files from the same directory may be combined using the **-path** option. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The pathname and file names must be enclosed in parentheses (for example, **-path (/opt/dcelocal/var/dfs/ admin.ft admin.fl)**).

If the **-path** option names only a single file from a directory an **upclient** process can request and receive only that file. An **upclient** process that requests the entire directory in which the file resides receives no files. If you specify multiple files and directories, the paths must be disjoint (nonintersecting). Paths are examined from left to right; any path that intersects with a previous path is logged as an error and ignored.

The **upserver** process writes error messages to the **dcelocal/var/dfs/adm/UpLog** event log file. When the **upserver** process is started, it creates the **UpLog** file if the file does not already exist. It then appends messages to the file. If the file exists when the **upserver** process is started, the process moves it to the **UpLog.old** file in the same directory (overwriting the current **UpLog.old** file if it exists) before creating a new version to which to append messages.

Only one **upserver** process should be run on a machine at one time. The **upserver** process automatically creates the **/opt/dcelocal/var/dfs/admin.up** file if the file does not already exist. A machine must be named in the **admin.up** file for its **upclient** process to be permitted to access files from the **upserver** process.

Privilege Required

You must be logged in as **root** on the local machine. On OS/390 DFS, **root** refers to a user with a **UID = 0**.

Implementation Specifics

On OS/390 DFS, the **upserver** command runs in the DFS server address space under the control of the **bosserver** process. The **bos create** command allows you to add and configure the **upserver**.

The **upserver** command is not supported on OS/390 DFS.

For further information about adding or configuring the **upserver**, refer to "bos create" on page 485.

When the **bos create** command is issued on OS/390 DFS, requests for different files from the same directory may be combined using the **-path** option. The trailing slash (/) after the pathname must be included before the file name. The full pathname must be specified, followed by the names of the requested files (separated by spaces). The pathname and file names must be enclosed in parentheses

In the following example, parameters are entered identifying the File Server machine **wichita** and the files the **upserver** process is to distribute. The **-path** option specifies the files to be distributed. Note that the full pathname, **(/opt/dcelocal/var/dfs/ admin.up admin.bos admin.bak admin.ft admin.fl)** is entered and that the pathname is surrounded by parentheses. In this example, the **admin.up**, **admin.bos**, **admin.bak**, **admin.ft**, and the **admin.fl** files are specified and separated by spaces. All reside in the same directory. These are files containing administrative lists for the Update Server (**upserver**), the Basic OverSeer Server (**bosserver**), the Backup Server (**bakserver**), the Fileset Server (**ftserver**), and the Fileset Location Server (**flserver**) processes.

```
$ bos create /.:/hosts/DCEDFS7 upserver simple              \
  "upserver envar('_EUV_HOME=/opt/dfslocal/home/upserver')/   \
  >dd:upserver 2>&1                                           \
  -path (/opt/dcelocal/var/dfs/ admin.up admin.bos admin.bak  \
  admin.ft admin.fl)"
```

Related Information

Commands:

bos create	**upclient**

Files:

admin.up	**UpLog**

Chapter 20. Distributed File Service Application Programming Interface

This chapter describes Programming Interface information. It describes the DFS application programming interface (API), **w_pioctl**.

w_pioctl - Pathname I/O Control

Purpose

Retrieves the contents of a DFS mount point.

Format

```
w_pioctl(Pathname_length,
         Pathname,
         Command,
         Argument_length,
         Argument,
         Return_value,
         Return_code,
         Reason_code)
```

Parameters

Pathname_length
An integer that contains the length of the pathname of the parent directory of the DFS mount point.

Pathname
A character string that contains the length of the pathname of the parent directory of the DFS mount point.

Command
An integer that contains the command to retrieve the DFS mount point. The command should contain the value X'4000C18E'.

Argument_length
An integer that contains the length of the argument. It should be the greater of:

- The length of the DFS mount point name plus the length of an integer (4), or
- 117 (the largest DFS mount point contents (113) plus the length of an integer (4)).

Note: The mount point contents is a DFS fileset name. The maximum length fileset name is 102 characters. This is augmented (optionally) by an extension (for example, .readonly) making 111 characters. A fileset name in a mount point is prefixed by a mount point type indicator (for example, #). Finally, a terminating null character is provided resulting in a maximum of 113 characters to be returned in the mount point contents.

The argument length cannot be greater than 1024.

Argument
A structure that has two areas used for both input and output.

For input (supplied by the caller of **w_pioctl**):

```
int      ┌──────────────────────────────────┐
         │length of the DFS mount point name│
         ├──────────────────────────────────┤
char []  │DFS mount point name              │
         │           .                      │
         │           .                      │
         │           .                      │
         └──────────────────────────────────┘
```

For output (returned by **w_pioctl**):

```
int       length of the DFS mount point contents

char []   DFS mount point contents
                    .
                    .
                    .
```

Return_value
An integer that contains 0 if the request is successful or -1 if it is not successful.

Return_code and **Reason_code**
An integer in which the return code or reason code is stored. A return code or reason code is only stored if the return value is -1. In addition to the return codes documented in the *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*, SC28-1899 for **w_pioctl**, see the following table.

Return_code Decimal Value	Return_code Hex Value	Return_code	Reason_code (Hex)	Description
121	0079	EINVAL	6C04000D	Invalid input buffer specification
			6C04000E	Argument is not a DFS mount point
122	007A	EIO	6C04000E	An I/O error occurred
128	0080	ENODEV	6C04000B	Command is invalid
129	0081	ENOENT	6C04000E	Internal System error
132	0084	ENOMEM	6C040005	Not enough virtual memory available
			6C040011	Not enough virtual memory available
135	0087	ENOTDIR	6C04000E	Pathname is not a directory
145	0091	E2BIG	6C04000E	Fileset name too large for output field
157	009D	EMVSERR	6C040009	Environmental or Internal System error
			6C040007	Environmental or Internal System error
1122	0462	ENOBUFS	6C040010	Insufficient output buffer space available

Privilege Required

The issuer's DCE principal or DCE group must have read permission on the directory that is the DFS mount point.

Examples

```
#include <stdio.h>
#include <string.h>
#include <bpxyvfsi.h>

#define  PATHSIZE 1024
#define  DIRSIZE   255

  char  pathnameP[PATHSIZE+1];       /* Parent pathname                    */
  long  command = 0x4000C18E;        /* Read contents of dfs mount point       */
  struct {
          int  len;                  /* Input  - length of mount point         */
                                     /* Output - length of mount point contents */
          char data_area[DIRSIZE+1]; /* Input  - mount point                   */
                                     /* Output - mount point contents          */
         } argument;
  int   Return_value = 0;
  int   Return_code  = 0;
  int   Reason_code  = 0;


int main(argc, argv)
        int     argc;
        char    *argv[];
{
  if (argc > 2)    /* if caller included at least a parent pathname and a mount point */
    {
     printf("Parent path = %s, Mount Point = %s\n", argv[1], argv[2]);
     strncpy(pathnameP,argv[1],PATHSIZE+1);        /* Copy parent path into pathname */
     strncpy(argument.data_area,argv[2],DIRSIZE+1); /* Copy mount point into argument */
     argument.len = strlen(argv[2]);    /* Length of mount point without null char */

            w_pioctl(strlen(pathnameP),
             pathnameP,
             command,
             (int)sizeof(argument),
             (char *)&argument,
             &Return_value,
             &Return_code,
             &Reason_code
             );

     printf("Return_value = %d, Return code = %d, Reason code = %X\n",
                          Return_value,     Return_code,      Reason_code);
     if (Return_value == 0)
       {
        printf("Mount Point contents for %s/%s is %s\n",argv[1],argv[2],argument.data_area);
        printf("Mount point contents length = %d\n",argument.len);
       }
    }
  else
    printf("Specify a parent path as 1st parameter and a mount point as 2nd parameter\n");
  return(Return_value);
}
```

Appendix A. Environment Variables in DFS

Environment variables affect the behavior of the DFS components. In DFS, the **envar** file of a client or server process can be created to contain the declarations of the environment variables.

The DCE environment variables affect the administration of DFS. This appendix describes the DCE and DFS environment variables DFS server processes and DFS client processes use, and tells you how to set them.

Environment variables are set using the following syntax:

VARIABLE_NAME=*value*

Environment Variables

Table 19 lists the OS/390 DCE and OS/390 DFS environment variables with their descriptions. See the *OS/390 DCE Administration Guide* for a complete list of DCE environment variables. This table also includes the Distributed File Service environment variables specific for SMB support. You can refer to the *OS/390 Distributed File Service SMB Administration Guide and Reference* for more information.

Notes:

1. Setting these environment variables is optional. In most cases, the default values are used if environment variables are not set.
2. In the following table all the examples should be entered on one line, with no blanks, even though some of them appear on multiple lines.

Table 19 (Page 1 of 16). Environment Variables

Name	Description
DCE Environment Variables	
_EUV_AUTOLOG	Determines if single sign-on processing is ignored when an OS/390 user invokes an application that invokes the DFS client. The valid value is: **NO** Do not enable single sign-on processing. Any other value causes DCE autologin processing to be attempted. **Note:** If _EUV_AUTOLOG=NO is specified, DCE autologin processing does not occur if the DFS namespace is accessed before a **dce_login** is performed. Refer to the *OS/390 DFS Administration Guide and Reference* for more information on DFS client and DFS autologin.
_EUV_ECHO_STDIN	Used by the RPC, CDS, and DTS control programs, as well as the Registry Editor and ACL Editor to echo input commands to the standard output file when these commands are run in batch. The default value is **0** (disabled).
_EUV_ENVAR_FILE	Defines the name of the environment variable file. The default value is *$HOME*/**envar**, where *$HOME* is your home directory.
_EUV_FTRACE	Activates function tracing within DCE or user code compiled with the TEST(NONE) compiler option.
_EUV_HOME	Used to override the home directory value that was specified in the POSIX segment of a RACF user ID. The default value is your home directory.
_EUV_SEC_KRB5CCNAME_FILE	Specifies the file that contains the declaration of the KRB5CCNAME variable. The default value is *$HOME*/**krb5ccname**.

Table 19 (Page 2 of 16). Environment Variables

Name	Description
_EUV_SVC_DBG_MSG_LOGGING	Turns debug tracing off.
_EUV_SVC_MSG_LEVEL	A DCE environment variable that sets the minimum severity level of messages that are displayed. For the Distributed File Service, the value **VERBOSE** is recommended. This is the default value if this **envar** file is not specified. For more information on this environment variable see the *OS/390 DCE Administration Guide*.
_EUV_SVC_MSG_LOGGING	A DCE environment variable used to control where messages are routed and displayed. For the Distributed File Service, the value **CONSOLE_LOGGING** is recommended and supplied in the sample **envar** file for each of the Distributed File Service processes. For more information on this environment variable see the *OS/390 DCE Administration Guide*.
BIND_PE_SITE	Determines whether the Security Server is located using the namespace or by reading the **pe_site** file. The default value is **0**, meaning the Security server is located by using a regular CDS query.
DCE_START_SOCKET_NAME	The path used as the well-known socket name by the DFS servers during daemon initialization. **Default Value** The default value for the DFS server is **/opt/dfslocal/home/dfskern/ioepk.soc** **Expected Value** Character string. **Example** DCE_START_SOCKET_NAME= /opt/dfslocal/home/dfscntl/start.sock **Where Variable is Used** The DFS server daemons.
KRB5CCNAME	Specifies the name of the user's Security credentials cache file. This variable is automatically managed within the DCE library and should not be set by the user. In OS/390 DFS, KRB5CCNAME is not an actual environment variable. However, as KRB5CCNAME is an environment variable in other implementations of DCE, it is included here for reference purposes.
NLSPATH	A POSIX environment variable used by DCE that sets the search path for message catalogs. The default value is: **/usr/lib/nls/msg/En_US.IBM-1047/%N: /usr/lib/nls/msg/%L/%N: /usr/lib/nls/msg/prime/%N**
SVC_CDS_DBG	Sets the CDS component debug level.
SVC_DTS_DBG	Sets the DTS component debug level.
SVC_PLT_DBG	Sets the Platform component debug level.
SVC_RPC_DBG	Sets the RPC component debug level.
SVC_SEC_DBG	Sets the Security component debug level.
TZ	Sets the time zone used by DTS. The default value is **localtime**.

DFS Environment Variables

Table 19 (Page 3 of 16). Environment Variables

Name	Description
_IOE_BACKUP_TAPE_CAPACITY	Specifies the size, in bytes, of the maximum size backup tape data set that is created during backup. It is desirable that this value equal the actual capacity of your tape volumes (or a little less). **Default Value** 160M (the size of a 3480 tape cartridge) **Expected Value** Size in bytes, of the maximum size backup tape data set. The value may be a whole positive integer followed by a 'K' (denoting kilobytes), or a whole positive integer followed by an 'M' (denoting megabytes). **Example** _IOE_BACKUP_TAPE_CAPACITY=180M **Where Variable is Used** BUTC*nn* (valid entries for *nn* are 01 through 08)
_IOE_BOSERVER_ENVAR	Specifies where to find the environment variable file during a **boserver** restart. **Default Value** ENVAR('_EUV_HOME=/opt/dfslocal/home/boserver') **Expected Value** Character string. **Example** _IOE_BOSERVER_ENVAR=ENVAR('__EUV_HOME= /opt/dfslocal/home/boserver') **Where Variable is Used** BOSERVER
_IOE_BOSERVER_OUTPUTDD	The data definition name (ddname), statically allocated in the DFS JCL procedure, which will receive the **boserver** output (before it switches to its log file). **Default Value** BOSERVER **Expected Value** Character string, eight characters or less. **Example** _IOE_BOSERVER_OUTPUTDD=BOSRV **Where Variable is Used** BOSERVER
_IOE_BOSERVER_PGMNAME	The OS/390 PDS member name of the **boserver** program to be used when restarting the **boserver** process. **Default Value** BOSERVER **Expected Value** Character string, eight characters or less. **Example** _IOE_BOSERVER_PGMNAME=NEWBOSRV **Where Variable is Used** BOSERVER
_IOE_BUTC_DISKUNIT	The unit name to be used when dynamically allocating a data definition for disk data sets. This is the unit name passed to dynamic allocation when allocating the disk device. **Default Value** SYSDA **Expected Value** Character string, six characters or less. **Example** _IOE_BUTC_DISKUNIT=3390 **Where Variable is Used** BUTC*nn* (valid entries for *nn* 01 through 08).

Table 19 (Page 4 of 16). Environment Variables

Name	Description
_IOE_BUTC_DUMP_HLQ	The high level qualifier for the dump (backup) data set names. **Default Value** DFSBKUP **Expected Value** Character string, 17 characters or less. **Example** _IOE_BUTC_DUMP_HLQ=DFSBKUP.VER1 **Where Variable is Used** BUTC*nn* (valid entries for *nn* 01 through 08).
_IOE_BUTC_KSDS	The name of the virtual storage access method (VSAM) DFS tape backup management file. **Default Value** DFSBUTC.KSDS.LABEL **Expected Value** Character string, 44 characters or less. **Example** _IOE_BUTC_KSDS=DFSBKUP.KSDS.LABEL **Where Variable is Used** BUTC*nn* (valid entries for *nn* 01 through 08).
_IOE_BUTC_TAPEUNIT	The unit name to be used when dynamically allocating a data definition for tape drives. This is the unit name passed to dynamic allocation when allocating the tape device. **Default Value** 3490 **Expected Value** Character string, six characters or less. **Example** _IOE_BUTC_TAPEUNIT=CART **Where Variable is Used** BUTC*nn* (valid entries for *nn* 01 through 08).
_IOE_CFG_CELL_ID	Specifies the DCE principal name of the administrator who is performing the OS/390 DFS configuration. **Default Value** cell_admin **Expected Value** Character string, 132 characters or less. **Example** _IOE_CFG_CELL_ID=cell_admin **Where Variable is Used** DFSCONF
_IOE_CFG_INFORM_LEVEL	Specifies the level of information that is displayed on the screen from the configuration of the host system. The valid values are: **0** Only messages regarding the progress of configuration are displayed. **1** Progress messages and the commands used by the configuration written. This is the default value. **2** Progress messages, command, and the output from most commands are written. **Default Value** 1 **Expected Value** 0, 1, or 2. **Example** _IOE_CFG_INFORM_LEVEL=2 **Where Variable is Used** DFSCONF

Table 19 (Page 5 of 16). Environment Variables

Name	Description
_IOE_CFG_LOG_FILE	Specifies the name of the configuration log file. **Default Value** dfsconf.log and is initially located in the home directory of the user running DFSCONF. **Expected Value** Character string, 132 characters or less. **Example** _IOE_CFG_LOG_FILE=/home/dfsadmin/dfsconf.log **Where Variable is Used** DFSCONF
_IOE_CM_ATTRIBUTES_FILE	Specifies the name of the file that contains the definition of file name suffixes that controls whether file data should be translated by the DFS client. **Default Value** None **Expected Value** Character string, 132 characters or less. **Example** _IOE_CM_ATTRIBUTES_FILE=/etc/httpd.conf **Where Variable is Used** DFS client processes **Notes** This file contains AddType statements in the same format as the Domino Go Webserver's httpd.conf file. All statements other than AddType are ignored. IBM supplies an example file in /opt/dfsglobal/examples/cmattr. This file can be copied and modified appropriately.
_IOE_CM_DIRCACHE_SIZE	Specifies the size, in bytes, of the directory cache for the DFS client. **Default Value** 1M **Expected Value** Size, in bytes, of the cache. The value may be a whole positive integer followed by a 'K' (denoting kilobytes), or a whole positive integer followed by an 'M' (denoting megabytes). **Example** _IOE_CM_DIRCACHE_SIZE=2M **Where Variable is Used** DFS client processes. **Notes** The minimum allowable size of the DFS client directory cache is one megabyte. This value will be used for the directory cache even if a lower value is specified. The maximum size is limited only by the virtual storage size of DFSCM. Values for the directory cache size will be rounded upward to a 128K boundary.
_IOE_CM_LOGIN_PROCESSES	The DFS environment variable controls the number of **ioelogin** processes set up to handle DCE login requests. **Default Value** 1 **Expected Value** A number greater than zero. **Example** _IOE_CM_LOGIN_PROCESSES=2 **Where Variable is Used** DFS client processes.

Table 19 (Page 6 of 16). Environment Variables

Name	Description
_IOE_CM_PARMS	The DFS client initialization parameters or options that take effect when the DFS client is started. The possible parameters are a subset of the **dfsbind** and the **dfsd** command options described in the *OS/390 DFS Administration Guide and Reference*. **Default Value** For the default values, see the **dfsbind** and **dfsd** command options in the "Commands" section of the *OS/390 DFS Administration Guide and Reference*. **Expected Value** Character string of DFS client initialization parameters (up to 1024 characters in length). **Example** `_IOE_CM_PARMS=-mountfilesystem IOE_DFS_CLIENT_DATA -cachedir 11` **Where Variable is Used** DFS client processes.
_IOE_CM_REQUEST_THREADS	The DFS environment variable controls the number of **DFSCM ioedfsd** process threads set up to handle file requests. **Default Value** 5 **Expected Value** A number greater than zero. **Example** _IOE_CM_REQUEST_THREADS=7 **Where Variable is Used** DFS client processes.
_IOE_DAEMONS_IN_AS	The DFS environment variable controls whether the DFSKERN process runs in its own address space or in the DFS Server Address Space. **Default Value** The default is ''. If '' is specified or the environment variable is not specified, DFSKERN runs in the DFS Server Address Space. **Expected Value** DFSKERN or '' **Example** _IOE_DAEMONS_IN_AS=DFSKERN In this case, DFSKERN runs in its own address space. **Where Variable is Used** DFSCNTL
_IOE_DFS_MODIFY_PATH	The path used as the well-known socket name by the DFS programs when registering with DFSCNTL to receive modify commands (F DFS,QUERY DFSKERN). **Default Value** The default value of the DFS server is **/opt/dfslocal/home/dfscntl/modify.rendezvous** **Expected Value** Character string. **Example** _IOE_DFS_MODIFY_PATH= /opt/dfslocal/home/dfscntl/test.rendezvous **Where Variable is Used** All programs. If the default is not being used, this environment variable must be coded in the **envar** file for each process.

Table 19 (Page 7 of 16). Environment Variables

Name	Description
_IOE_DIRECTORY_CACHE_SIZE	The number of 512 byte blocks used to cache HFS and RFS directory entries. **Default Value** 768 **Expected Value** The value specified must be a numeric value greater than or equal to 768. **Example** _IOE_DIRECTORY_CACHE_SIZE=1024 **Where Variable is Used** DFSKERN
_IOE_EPI_CACHE_SIZE	The size, in bytes, of the DCE Local File System metadata buffer cache. This environment variable is used to increase the size of the buffer cache and to potentially improve performance. **Default Value** one megabyte. This value is also the minimum value. **Expected Value** Size, in bytes, of the cache. The value expected may be a whole positive integer followed by a 'K' (denoting kilobytes), or a whole positive integer followed by a 'M' (denoting megabytes). **Example** _IOE_EPI_CACHE_SIZE=8M **Where Variable is Used** DFS client processes and DFSKERN **Notes** The minimum allowable size of the DCE Local File System buffer cache is one megabyte. This value will be used for the buffer cache even if a lower value is specified. The maximum size is limited only by the virtual storage size of DFSKERN. Values for the buffer cache size will be rounded upward to a 128K boundary.
_IOE_HFS_ATTRIBUTES_FILE	Specifies the pathname of the **hfsattr** file that contains the definition of file name suffixes that controls whether file data should be translated by the DFS server. **Default Value** None. **Expected Value** Character string, 132 characters or less. **Example** _IOE_HFS_ATTRIBUTES_FILE=/etc/httpd.conf **Where Variable is Used** DFSKERN **Notes** This file contains AddType statements in the same format as the IBM HTTP Server's http.conf file. All statements other than AddType are ignored. IBM supplies an example file in **/opt/dfsglobal/examples/cmattr**. This file can be copied and modified appropriately. (Only one example file is provided for **cmattr** and **hfsattr** since they are the same format.)

Table 19 (Page 8 of 16). Environment Variables

Name	Description
_IOE_HFS_TRANSLATION	A DFS server variable that controls the conversion of HFS data to the appropriate data format. Converts incoming data from ASCII ISO 8859-1 to the local OS/390 code page. Converts outgoing data from the local OS/390 code page to ASCII ISO 8859-1. **Default Value** Off **Expected Value** On, Off, or Auto **On** means that all data will be translated. **Off** means that no data will be translated. **Auto** means that data that is received by (read) or sent to (written) the DFS server is examined to determine if the data is text or not. When data is received by (read) the DFS server, the first 255 bytes of data are compared against a table of valid text (EBCDIC) characters. If all 255 characters are deemed to be valid text characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated. When data is sent from (written) the DFS server, the first 255 bytes of data are compared against a table of valid text (ASCII) characters. If all 255 characters are deemed to be valid text characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated. **Example** _IOE_HFS_TRANSLATION=ON **Where Variable is Used** DFSKERN **Notes** For HFS, the following sequence is used by the DFS server to determine whether to translate file data: If the file exists and has a non-zero file format attribute (ST_FILEFMT) field in the BPXYSTAT structure, then a value of 1 (FTFFBINARY) causes no translation to be done. A value greater than 1 causes translation to be done. If the file does not exist or has a zero file format attribute, then if an **hfsattr** file is specified (by the DFSKERN _IOE_HFS_ATTRIBUTES_FILE environment varible), and the file name suffix matches a directive in the **hfsattr** file, then that controls whether translation occurs. The fileset's **devtab** translation parameter (**text**, **binary**, or **auto**) is used when no **hfsattr** file is specified, or if the file name suffix does not match any of the directives in the **hfsattr** file, or the file name has no suffix. The DFSKERN _IOE_HFS_TRANSLATION environment variable is used if the fileset's **devtab** entry does not have a translation parameter. The ST_FILEFMT field in the BPXYSTAT structure for HFS files is set if it is not already set and either an **hfsattr** file was used to determine whether to translate or **auto** (on the fileset or globally) was used to determine whether to translate. This variable does not apply to OS/390 DFS client processing. DFS client translation is controlled by the **dfsd** command **-translation** option described in the *OS/390 DFS Administration Guide and Reference*.

Table 19 (Page 9 of 16). Environment Variables

Name	Description
_IOE_LFS_SYNC_INTERVAL	The number of seconds between synchronization operations for the DCE Local File System. **Default Value** 30 **Expected Value** Numeric, greater than zero. **Example** _IOE_LFS_SYNC_INTERVAL=60 **Where Variable is Used** DFSKERN
| _IOE_MVS_DFSDFLT	The name of the RACF-defined anonymous user that is associated with unauthenticated DCE users attempting access to exported HFS files. This ID must be RACF-defined with an OS/390 UNIX segment. **Default Value** There is no default value for this variable. **Expected Value** Character string, eight characters or less. **Example** _IOE_MVS_DFSDFLT=DFSDFLT **Where Variable is Used** DFSKERN (used when exporting HFS file systems).
_IOE_MVS_IDMAP	The name of the identity mapping output file that establishes the relationship between DCE user IDs and OS/390 user IDs (Resource Access Control Facility (RACF)). **Default Value** There is no default value for this variable. **Expected Value** Character string describing the map file to be used. **Example** _IOE_MVS_IDMAP=/opt/dfslocal/home/dfskern/mapid.out **Where Variable is Used** DFSKERN (used when exporting HFS file systems).
_IOE_MVS_IDMAP_SAF	The DFS Server variable establishes whether a System Access Facility (such as, RACF) is used for UUID to OS/390 ID mapping (YES) or if the identity mapping output file is used (NO). If YES is specified, the _IOE_MVS_IDMAP environment variable is ignored. **Default Value** NO **Expected Value** NO or YES **Example** _IOE_MVS_IDMAP_SAF=YES **Where Variable is Used** DFSKERN
_IOE_MVS_SERVER	The name of the OS/390 server used for identity mapping and registering with OS/390 UNIX as a DFS server. **Default Value** DFS **Expected Value** Character string, maximum 32 characters. **Example** _IOE_MVS_SERVER=MVS/DFS01 **Where Variable is Used** DFSKERN

Table 19 (Page 10 of 16). Environment Variables

Name	Description
_IOE_PROTOCOL_RPC	An environment variable that controls whether the DCE DFS protocol is supported (using DCE RPC). **Default Value** OFF **Expected Value** ON or OFF **Example** _IOE_PROTOCOL_RPC=OFF **Where Variable is Used** DFSKERN
_IOE_PROTOCOL_SMB	An environment variable that controls whether the SMB protocol is supported (using TCP/IP). **Default Value** OFF **Expected Value** ON or OFF **Example** _IOE_PROTOCOL_SMB=ON **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_RFS_ATTRIBUTES_FILE	The name of the file that contains the table for describing the attributes used to manipulate record files in the DFS Server. The value can be the name of the HFS file, a fixed-block partitioned data set, or a fixed-block sequential data set with a record length of 80. **Default Value** /opt/dfslocal/var/dfs/rfstab **Expected Value** Character string describing the attributes file being used (maximum 255 characters). **Example** _IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)' **Where Variable is Used** DFSKERN **Notes** This DFS server variable can be overridden for specific filesets in the **devtab** file. For more information, see the *OS/390 DFS Administration Guide and Reference*.
_IOE_RFS_STATUS_REFRESH_TIME	The DFS Server refreshes its cache of exported record data set names and attributes at a regular interval. The time is specified in seconds. **Default Value** 600 seconds (10 minutes) **Expected Value** A number greater than zero. **Example** _IOE_RFS_STATUS_REFRESH_TIME=360 **Where Variable is Used** DFSKERN

Table 19 (Page 11 of 16). Environment Variables

Name	Description
_IOE_RFS_TRANSLATION	A DFS server variable that controls the conversion of RFS data to the appropriate data format. Converts incoming data from ASCII ISO 8859-1 to the local OS/390 code page. Converts outgoing data from the local OS/390 code page to ASCII ISO 8859-1. **Default Value** Off **Expected Value** On or Off **Example** _IOE_RFS_TRANSLATION=ON **Where Variable is Used** DFSKERN **Notes** This DFS server variable can be overridden for specific filesets in the **devtab** file. If an individual fileset is set to binary, in the **devtab** file, no translation will be performed. For more information, see the *OS/390 DFS Administration Guide and Reference*. This variable does not apply to OS/390 DFS client processing. DFS client translation is controlled by the **dfsd** command **-translation** option described in the *OS/390 DFS Administration Guide and Reference*.
_IOE_RFS_WORKER_THREADS	A DFS server variable that controls the number of threads to be started in **dfskern** to service open and close requests for RFS files. **Default Value** 1 **Expected Value** A number greater than zero. **Example** _IOE_RFS_WORKER_THREADS=3 **Where Variable is Used** DFSKERN
_IOE_SERVER_NAME	The network name of the system. This environment variable should be used if there are several network connections on the OS/390 system and the primary connection is not being used for the DFS system. **Default Value** The name that is returned from the TCP/IP function, **gethostname()**. The name is the default name as determined from the TCP/IP **TCPDATA** file and may not reflect the actual network name for the device being used for the DFS network. **Expected Value** Character string containing the network name of the system without a domain (note that TCP/IP returns the name in upper case so this name should be in upper case also). **Example** _IOE_SERVER_NAME=RANDOM **Where Variable is Used** All programs that use the TCP/IP function, **gethostname()**.
| _IOE_SMB_BROWSE_INTERVAL	Specifies the Browser announcement interval (in milliseconds). **Default Value** 72000 (1.2 minutes) **Expected Value** A number between 0 and 720000 (12 minutes). **Example** _IOE_SMB_BROWSE_INTERVAL=90000 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.

Table 19 (Page 12 of 16). Environment Variables

Name	Description
_IOE_SMB_CALLBACK_POOL	Specifies the number of secondary pool threads for processing SMB callback requests. **Default Value** 2 **Expected Value** A number greater than 0. **Example** _IOE_SMB_CALLBACK_POOL=3 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_COMPUTER_NAME	Specifies the name to be used by SMB redirectors (that is, clients) to contact this server. If a Windows Internet Naming Service (WINS) server is available (see the _IOE_SMB_PRIMARY_WINS environment variable), the SMB computer name is used to identify this server to the WINS server. **Default Value** The TCP/IP hostname of this system. **Expected Value** Character string, 15 characters or less. **Example** _IOE_SMB_COMPUTER_NAME=OS390DATA1 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_DESCRIPTION	Specifies the description of this server that appears on the PC. **Default Value** None **Expected Value** Character string, 40 characters or less. **Example** _IOE_SMB_DESCRIPTION=OS390 File Server Note in this example, there are embedded blanks. **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_DOMAIN_NAME	Specifies the name to be used as the Domain name for this server. **Default Value** None **Expected Value** Character string, 15 characters or less. **Example** _IOE_SMB_DOMAIN_NAME=OS/390DOMAIN1 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.

Table 19 (Page 13 of 16). Environment Variables

Name	Description
_IOE_SMB_IDMAP	Specifies the location of the **smbidmap** file. The **smbidmap** file contains the mapping of SMB user IDs to OS/390 user IDs. **Default Value** None **Expected Value** Character string specifying the path name of the **smbidmap** file. **Example** _IOE_SMB_IDMAP=/opt/dfslocal/home/dfskern/smbidmap **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_IDLE_TIMEOUT	Specifies how long (in seconds) an SMB session can remain inactive before it is terminated. **Default Value** 400 **Expected Value** A number between 0 and 4294967295. **Example** _IOE_SMB_IDLE_TIMEOUT=4000 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_MAIN_POOL	Specifies the number of primary pool threads for processing SMB requests. **Default Value** 14 **Expected Value** A number greater than 0. **Example** _IOE_SMB_MAIN_POOL=20 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_MAXXMT	Specifies the maximum server buffer size that is returned on the SMB Server Negotiate response. **Default Value** 4356 bytes **Expected Value** A number between 1024 and 65535. **Example** _IOE_SMB_MAXXMT=8192 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_OPLOCK_TIMEOUT	Specifies the Opportunistic Lock Timeout period (in seconds) **Default Value** 300 **Expected Value** A number between 0 and 4294967295. **Example** _IOE_SMB_OPLOCK_TIMEOUT=60 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.

Table 19 (Page 14 of 16). Environment Variables

Name	Description
_IOE_SMB_PRIMARY_WINS	Specifies the IP address of the Windows Internet Naming Service (WINS) server that this server announces itself to and forwards WINS requests to. **Default Value** None **Expected Value** An IP address (*n.n.n.n* where *n* is a number between 0 and 255). **Example** _IOE_SMB_PRIMARY_WINS=9.120.44.55 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_RAW	Specifies whether raw mode is supported in the SMB Server Negotiate response. **Default Value** ON **Expected Value** ON or OFF **Example** _IOE_SMB_RAW=OFF **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_SCOPE	Specifies the Scope ID for the Windows Internet Naming Service (WINS) server. The Scope ID defines a group of computers that recognize a registered NetBOIS name. (This should normally be omitted.) **Default Value** None **Expected Value** Character string, 224 characters or less. **Example** _IOE_SMB_SCOPT=MYDEPARTMENTSCOPE **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_SECONDARY_WINS	Specifies the IP address of the Windows Internet Naming Service (WINS) server that this server announces itself to and forwards WINS requests to if the Primary WINS server does not respond. **Default Value** None **Expected Value** An IP address (*n.n.n.n* where *n* is a number between 0 and 255). **Example** _IOE_SMB_SECONDARY_WINS=9.120.66.77 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.

Table 19 (Page 15 of 16). Environment Variables

Name	Description
_IOE_SMB_TOKEN_FILE_MAX	Specifies the maximum number of files that the SMB token cache should keep tokens for. **Default Value** 1024 **Expected Value** A number greater than or equal to 1024. **Example** _IOE_SMB_TOKEN_FILE_MAX=2048 **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_SMB_WINS_PROXY	Specifies whether this server can act as a Windows Internet Naming Service (WINS) server proxy. WINS requests sent to this server will be forwarded to a WINS server on another computer (see the _IOE_SMB_PRIMARY_WINS environment variable). **Default Value** OFF **Expected Value** ON or OFF. **Example** _IOE_SMB_WINS_PROXY=ON **Where Variable is Used** DFSKERN **Note** This is used for SMB support only.
_IOE_TKM_MAX_TOKENS	Specifies the maximum number of tokens that can be held in the DFS server memory. **Default Value** 1024 **Expected Value** A positive number. The minimum value is 1024. **Example** _IOE_TKM_MAX_TOKENS=2048 **Where Variable is Used** DFSKERN
_IOE_TKMGLUE_SERVER_THREADS	The number of threads to be started in **dfskern** to service token requests from the glue layer. **Default Value** 5 **Expected Value** The value specified must be a numeric value greater than or equal to 5. **Example** _IOE_TKMGLUE_SERVER_THREADS=8 **Where Variable is Used** DFSKERN

Table 19 (Page 16 of 16). Environment Variables

Name	Description
_IOE_VM_CACHE_SIZE	Specifies the size, in bytes, of the virtual memory used by the DFS client and DFSKERN for the DCE Local File System. **Default Value** 1M **Expected Value** A positive number. The value may be a whole positive integer followed by a 'K' (denoting kilobytes), or a whole positive integer followed by an 'M' (denoting megabytes). **Example** _IOE_VM_CACHE_SIZE=2M **Where Variable is Used** Can be specified independently in the DFS client processes and DFSKERN **Notes** The virtual memory is used for data in the DCE Local File System. Metadata for the DCE Local File System is cached under control of the _IOE_EPI_CACHE_SIZE environment variable.
_IOE_VM_MAX_FILES	Specifies the maximum number of files that can be contained in the virtual memory used by the DFS client and DFSKERN for the DCE Local File System. **Default Value** 1024 **Expected Value** A positive number. The minimum value is 1024. **Example** _IOE_VM_MAX_FILES=2048 **Where Variable is Used** Can be specified independently in the DFS client processes and DFSKERN.
_IOE_VNODE_CACHE_SIZE	The size of the HFS and RFS vnode cache, that is, the number of vnodes. **Default Value** 4096 **Expected Value** The value specified must be a numeric value greater than or equal to 2048. **Example** _IOE_VNODE_CACHE_SIZE=6144 **Where Variable is Used** DFSKERN

Format for Setting Environment Variables

Environment variables are set using the following format:

variable-name=*value*

For example,

```
_IOE_MVS_DFSDFLT=DFSDFLT
```

There should be no space between the *variable-name* or the *value* and the equal sign that separates them. If a definition is too long to fit in a single line, you can continue the declaration to the next line by appending a backslash ("\") at the end of the current line. For example:

```
LONG_VARIABLE=this variable is far too long \
for one line.
```

How to Set Environment Variables

Use the **envar** file to set environment variables. You can override variables defined in the **envar** file from batch using the **envar** runtime option.

Setting Variables in the Environment Variable File

You can set environment variables by editing the **default environment variable file**, also known as the **envar** file. This file contains the environment variable declarations, one declaration per line (unless the declaration string straddles multiple lines due to length).

This file must be created in the user's home directory with the name **envar**. (You can change the pathname of the default environment variable file by setting the value of the **_EUV_ENVAR_FILE** environment variable to a specific pathname.)

Following is an example of the contents of the environment variable file:

```
BIND_PE_SITE=1
_EUV_ECHO_STDIN=1
RPC_DEFAULT_ENTRY=/.:/servers/server1
```

Environment variable declarations made through EXEC or CALL statements in TSO or batch (either by explicit declarations or pointing to an environment variable file) override the environment variable declarations in the default environment variable file while the program is executing. This is discussed in “Setting Environment Variables from Batch or TSO” on page 792.

Environment variables that are set through the **export** command in the shell environment override the default environment variable file while executing in the shell. This is discussed in “Setting Environment Variables from the Shell.”

If you do not declare environment variables or point to an environment variable file in an EXEC or CALL statement in TSO or batch, or if you do not declare an environment variable using the export command in the shell, the program will use the default environment variable file in your home directory (with the name **envar**). If this file does not exist, the program assumes that there is no environment variable file.

Setting Environment Variables from the Shell

You can use the export command to set environment variables from the shell.

In OS/390 UNIX, the export command sets a value for a variable and makes the variable available to the shell and to all processes forked by the shell. The common method of running this command is to include it in your **.profile** file which is run every time you enter the shell. For example, to set and export an environment variable, enter the following from the shell prompt or include it in your **.profile** file:

```
export BIND_PE_SITE=1
```

For more information on the export command or the **.profile** file, refer to the *OS/390 UNIX System Services User's Guide*, SC28-1891.

If you do not explicitly declare environment variables through the export command (from the command line or through the **.profile** file) in the shell, the default environment variable file (**envar** in your home directory) is used.

Setting Environment Variables from Batch or TSO

Environment variables can be set from batch or TSO by using the **envar** runtime option. These are described in the following sections.

If you do not explicitly declare environment variables using any of the methods described in this section, the default environment variable file (**envar**) is used.

Setting Environment Variables from Batch: In batch, the PARM statement contains the **envar** runtime options and the parameters that will be passed to the program. The "/" character separates the runtime options and the parameters. For example:

```
//PROG  EXEC PGM=PROG, PARM=('POSIX(ON),STACK(12000),ENVAR(''var1=value1'' ''var2=value2'')/pgm-parms')
```

Note: The **envar** declaration is enclosed in two single quotation marks, not double quotation marks.

The PARM statement is limited to 100 characters and if the **envar** and other PARM declarations exceed this limit, an alternative way must be used. The alternative is to point to a file (other than the default environment variable file) that contains all the variable declarations using the format:

```
...ENVAR(''_EUV_ENVAR_FILE=filename'')
```

where *filename* is the name of the HFS file or OS/390 data set that contains all the variable declarations, one declaration per line. If this is a relative HFS pathname, the file must exist in the user's home directory.

You can also refer to the environment variable file using the following format:

```
...ENVAR(''_EUV_ENVAR_FILE=//DD:filename'')...
```

where *filename* is the DD name of a DD statement that specifies the file containing the environment variables.

Setting Environment Variables from TSO: Environment variables can be set when CALLing a program through the **envar** runtime option using the following format:

```
CALL MYPROG 'POSIX(ON),STACK(12000),ENVAR(''var1=value1'' ''var2=value2'')/pgm-parms'
```

As in batch, **envar** can also be made to point to a file (other than the default environment variable file). For example:

```
CALL MYPROG 'POSIX(ON),ENVAR("_EUV_ENVAR_FILE=file1")/pgm-parms'
```

Again, the environment variable file is an HFS file or OS/390 data set.

DFS Daemon Environment Variable Files

DFSCNTL and each DFS daemon has its own home directory. Each home directory contains the **environment variable file** (**envar**), where the environment variables are set for each of the daemons.

The **envar** files of the DFS daemons are in:

- /opt/dfslocal/home/bakserver/envar
- /opt/dfslocal/home/boserver/envar
- /opt/dfslocal/home/butc01/envar
- /opt/dfslocal/home/butc02/envar
- /opt/dfslocal/home/butc03/envar
- /opt/dfslocal/home/butc04/envar
- /opt/dfslocal/home/butc05/envar
- /opt/dfslocal/home/butc06/envar

- /opt/dfslocal/home/butc07/envar
- /opt/dfslocal/home/butc08/envar
- /opt/dfslocal/home/daemonct/envar
- /opt/dfslocal/home/dfscm/envar
- /opt/dfslocal/home/dfscntl/envar
- /opt/dfslocal/home/dfsexport/envar
- /opt/dfslocal/home/dfskern/envar
- /opt/dfslocal/home/flserver/envar
- /opt/dfslocal/home/ftserver/envar
- /opt/dfslocal/home/growaggr/envar
- /opt/dfslocal/home/newaggr/envar
- /opt/dfslocal/home/repserver/envar
- /opt/dfslocal/home/salvage/envar
- /opt/dfslocal/home/upclient/envar
- /opt/dfslocal/home/upserver/envar

You can customize the values of the environment variables in these files to suit your operational needs.

Messaging Subsystem Environment Variables

DFS uses a DCE messaging facility that displays messages from DFS services. You can control the display of these messages based on the severity level using the **_EUV_SVC_MSG_LEVEL** environment variable. You can also control where the messages are displayed using the **_EUV_SVC_MSG_LOGGING** environment variable.

Important Note to Users

The DFS message catalogs must be put in a directory that is pointed to by the NLSPATH environment variable. If not, DFS will not be able to convert DFS error numbers to text messages. See the OS/390 Program Directory for more information on this.

Appendix B. Sharing Data Between OS/390 UNIX Applications, Commands, and DFS Clients

Note: In this appendix, UFS refers to a non-Local File System aggregate, that is, the HFS or RFS filesets.

Character data stored in the UNIX File System (UFS) filesets by DFS clients will be in ASCII format. OS/390 applications that assume that character data is EBCDIC will not interpret this data correctly. In addition, OS/390 EBCDIC data stored in UFS filesets will not be correctly interpreted by these clients as ASCII character format is assumed. If you want to share character data between DFS clients and OS/390 applications, use the **dfskern** process HFS translation environment variable, **_IOE_HFS_TRANSLATION**, or the RFS translation environment variable, **_IOE_RFS_TRANSLATION**.

The variable makes the UFS data appear in ASCII data format to the DFS clients. This function ensures that all outgoing character data is converted from the local code page to the ASCII page ISO 8859-1. Conversely, it ensures that all incoming character data is converted from ASCII ISO 8859-1 to the local OS/390 code page. Local OS/390 applications will still see UFS character data in EBCDIC format.

The default for the **_IOE_HFS_TRANSLATION** and **_IOE_RFS_TRANSLATION** environment variables is **OFF**. This means that no data is translated. If it is set to **ON**, all data written to UFS from DFS clients is translated from ASCII to EBCDIC. In addition, all data read from UFS by the DFS clients is translated from EBCDIC to ASCII. The translation function has no effect on DCE Local File System data.

Character data translation can also be specified on a UFS fileset basis. That is, one UFS fileset can have translation set to **ON** and another can have translation set to **OFF**. The HFS fileset translation control parameter overrides the **_IOE_HFS_TRANSLATION** environment variable setting for that HFS fileset and the RFS fileset translation control parameter overrides the **_IOE_RFS_TRANSLATION** environment variable setting for that RFS fileset. See page 157 for more information on specifying character data translation on an UFS fileset basis.

The ASCII code page is assumed to be ISO 8859-1. This is the code page used by DCE for character data. DFS does not allow you to specify the ASCII code page used. The EBCDIC code page used for UFS data is the local code page for the **dfskern** process. The default code page for OS/390 UNIX is IBM-1047.

Notes:

1. When the HFS or RFS translation environment variable is set to **ON**, the OS/390 DFS server assumes that the incoming data is ASCII code page ISO 8859-1 and outgoing data is ISO 8859-1. This may not be the case in your environment. DFS clients (other than the OS/390 DFS client, see page 590) do not do any translation prior to sending data out. However, many ASCII code pages are the same for letters, numbers and common punctuation. For example, ASCII code pages IBM-850 and ISO 8859-1 are the same for these characters:

```
       -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
   2- (sp) !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
   3-   0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
   4-   @  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
   5-   P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _
   6-   `  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
   7-   p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~
```

   It is possible that some of these characters will not print or will print incorrectly. The following are the hexadecimal values and their corresponding characters:

```
5B - left bracket
5C - back slash
5D - right bracket
5E - caret
60 - left quote
7B - left brace
7D - right brace
7E - tilde
```

2. Data sent from DFS clients may not be character data. For example, a DFS client could store an AIX (binary) executable on OS/390 in UFS. If the HFS or RFS translation environment variable is set to **ON**, binary data will be translated as if it were character data. Each byte value will be translated to another byte value successfully. When the DFS client retrieves this data from HFS or RFS, it will be translated back as if it were character data if there is a successful one-to-one mapping.

   Each time the DFS File Server is started, it checks the ISO 8859-1 code page against the local OS/390 code page for one-to-one mapping of every byte from ASCII to EBCDIC and back to ASCII. If the one-to-one mapping indicates that all bytes successfully map back to the original ASCII characters, the binary executable will be mapped back to its original form and it will execute properly. If the one-to-one mapping is unsuccessful, the DFS File Server displays the following warning message for HFS:

   ```
   IOEN00204A HFS Translation set ON, binaries may not translate correctly.
   ```

   or the following message for RFS:

   ```
   IOEN00204A RFS Translation set ON, binaries may not translate correctly.
   ```

   Note: RFS is not recommended for storing binary executables.

   Also, if you use the **get** or **put** subcommand of the TCP/IP **ftp** command to transfer a binary executable to UFS in binary mode and attempt to execute it from a DFS client (with the HFS or RFS translation environment variable turned **ON**), the program will not run. This is because the executable will be translated as character data by the OS/390 DFS server when retrieved by the DFS client.

   To avoid this problem, either store the executable using the DFS client or store it in the DCE Local File System where no translation is done.

3. If you have existing ASCII data in UFS and the HFS or RFS translation environment variable is set to **ON**, then you will not be able to retrieve the data properly.

   Once the HFS or RFS translation environment variable is set to **ON**, you should not turn it **OFF** (and vice versa). If you do, you will end up with some data stored in UFS that is ASCII and some that is EBCDIC.

   You can use the **dd** command and the **tr** command (available on AIX and other UNIX systems) to convert data from one character representation to another. For syntax on the **dd** command, see the *OS/390 UNIX System Services Command Reference*, SC28-1892.

   For example, if you want to run with the HFS or RFS translation environment variable **ON**, you can translate the ASCII data to EBCDIC from OS/390 by issuing the following commands from the OS/390 shell:

   ```
   $ dd if=asciidata conv=ebcdic | tr '\045' '\025' >ebcdicdata
   ```

Notes:

a. In the above example, the **dd** command converts the ASCII characters to EBCDIC. In addition, it converts the ASCII line feed character (X'0A') to X'25' (\045 in octal). The **tr** command changes X'25' to X'15' (\025 in octal) which is the EBCDIC line feed character. The output is redirected from the ASCII format file, **asciidata**, to the EBCDIC format file, **ebcdicdata**.

b. The above example also assumes that line feed is the end-of-line terminator for an RFS fileset. This is specified (defaulted) in the attributes file.

If you want to run with the HFS or RFS translation environment variable set to **OFF** (for example, if you do not want to share data with OS/390 applications), you can translate the EBCDIC data to ASCII from OS/390 by issuing the following commands from the OS/390 shell:

```
$ dd if=ebcdicdata conv=ascii | tr '\205' '\012' >asciidata
```

Notes:

a. In the above example, the **dd** command converts the EBCDIC characters to ASCII characters. In addition, it converts the EBCDIC line feed character (X'15') to X'85' (\205 in octal). The **tr** command changes X'85' to X'0A' (\012 in octal) which is the ASCII line feed character. The output is redirected from the EBCDIC format file, **ebcdicdata**, to the ASCII format file, **asciidata**.

b. The above example also assumes that line feed is the end-of-line terminator for an RFS fileset. This is specified (defaulted) in the attributes file.

4. When the HFS or RFS translation environment variable is set to **ON**, all data that is read or written from or to UFS by DFS clients is translated (unless you specify that character data translation should be done on a UFS fileset basis). This means that you cannot store data in UFS that should be translated, (character data, for example) and data that should not be translated, for example, binary data, that is interpreted or translated by the application itself, if you want to share both types of data with OS/390 applications. In this case, you must store the data to be translated in one UFS fileset and the data that is not to be translated in another UFS fileset. You must also set the translation control parameter appropriately for each of the UFS filesets (**text** for the translated UFS fileset, **binary** for the untranslated UFS fileset). See page 157 for more information on specifying character data translation on an UFS fileset basis.

5. Translation of DBCS data is not supported. File names, directory names, and the contents of symbolic links must be single byte character names.

Appendix C. Examples of Working with Character Data from Other Platforms

The following examples assume that the **_IOE_HFS_TRANSLATION** environment variable in the **dfskern** process is set to **OFF** (no translation) or the **devtab** entry for the HFS fileset being accessed is set to binary (no translation).

Using OS/390 HFS Data from AIX DFS Client Workstations

Data that was created by an OS/390 application will generally be in S/390, or EBCDIC, format. When data is stored in an HFS file with a file system that is exported by DFS, the data is made available to DFS client machines. In general, applications running on these client machines do not expect EBCDIC format data. Generally, client machines expect data in ASCII format.

You can use the **dd** command and the **tr** command (available on AIX and other UNIX systems) to convert data from one character representation to another. For syntax on the **dd** command, see the *OS/390 UNIX System Services Command Reference*, SC28-1892.

For example, to display an EBCDIC file (**hfsfile**) on an ASCII workstation, enter:

```
$ dd if=/.:/fs/hfsfile conv=ascii | tr '\205' '\n'
```

EBCDIC format data can be stored in a local workstation file in ASCII format. To store the contents of the EBCDIC format file **hfsfile** in the ASCII format file **asciifile**, enter:

```
$ dd if=/.:/fs/hfsfile conv=ascii | tr '\205' '\n' >asciifile
```

The ASCII format file can be modified and stored in EBCDIC format. If the **asciifile** is modified, it can be stored in the file **hfsfile** with:

```
$ dd if=asciifile conv=ebcdic | tr '\045' '\025' >/.:/fs/hfsfile
```

Note: In the first two examples, the **dd** command converts the EBCDIC characters to ASCII characters. In addition, it converts the EBCDIC line feed character (X'15') to X'85' (\205 in octal). The **tr** command changes X'85' to X'0A' (\012 in octal) which is the ASCII line feed character. In the last example, the **dd** command converts the ASCII characters to EBCDIC. In addition, it converts the ASCII line feed character (X'0A') to X'25' (\045 in octal). The **tr** command changes X'25' to X'15' (\025 in octal) which is the EBCDIC line feed character.

Using Data Stored by AIX DFS Client Workstations from OS/390

Data that was created by an application running on an AIX or other UNIX workstation will generally be in workstation, or ASCII, format. When data is stored in an HFS file with a file system that is exported by DFS, the data is made available to OS/390 applications. In general, applications running on OS/390 do not expect ASCII format data. Generally, applications running on OS/390 expect EBCDIC format data.

You can use the **dd** command and the **tr** command (available on the OS/390 shell) to convert data from one character representation to another. For syntax on the **dd** command, see the *OS/390 UNIX System Services Command Reference*, SC28-1892.

For example, to display an ASCII file (**asciidata**) on OS/390, enter:

```
$ dd if=asciidata conv=ebcdic | tr '\045' '\025'
```

ASCII format data can be stored in a local HFS file in EBCDIC format with the **dd** command. To store the contents of the ASCII format file **asciidata** in the EBCDIC format file **hfsdata**, enter:

```
$ dd if=asciidata conv=ebcdic | tr '\045' '\025' >hfsdata
```

The EBCDIC format file can be modified and stored in ASCII format. If **hfsdata** is modified, it can be stored back into the **asciidata** file with:

```
$ dd if=hfsdata conv=ascii | tr '\205' '\012' >asciidata
```

Note: In the first two examples, the **dd** command converts the ASCII characters to EBCDIC characters. In addition, it converts the ASCII line feed character (X'0A') to X'25' (\045 in octal). The **tr** command changes X'25' to X'15' (\025 in octal) which is the EBCDIC line feed character. In the last example, the **dd** command converts the EBCDIC characters to ASCII characters. In addition, it converts the EBCDIC line feed character (X'15') to X'85' (\205 in octal). The **tr** command changes X'85' to X'0A' (\012 in octal) which is the ASCII line feed character.

Using OS/390 HFS Data from OS/2 DFS Client Machines

Data that was created by an OS/390 application and stored in HFS can also be accessed from OS/2.

Two simple REXX programs can be used to convert data. The **eb2asc** program converts EBCDIC characters to ASCII characters. The **asc2eb** program converts ASCII characters to EBCDIC characters. (The source code of the REXX program used in this section is included in “Example REXX programs for OS/2 character conversion” on page 802. Your DFS Administrator can tell you if these programs are already available for your use.) For example, the following command displays an EBCDIC file (**hfsfile**) on your OS/2 machine (assuming that D: is where the DCE namespace **/...** is mounted):

```
[C:\] type d:\cellname\fs\hfsfile | eb2asc
```

EBCDIC data can be stored in a local OS/2 file in ASCII format. To store the contents of the EBCDIC format **hfsfile** file in a local OS/2 ASCII format file called **asciifil.txt**, enter:

```
[C:\] type d:\cellname\fs\hfsfile | eb2asc >asciifil.txt
```

The text of the OS/2 ASCII format file can be modified and stored in an EBCDIC file. If **asciifil.txt** is modified, it can be stored in the file **hfsfile** with:

```
[C:\] type asciifil.txt | asc2eb >d:\cellname\fs\hfsfile
```

Note: In the first two examples, the **eb2asc** REXX program converts the EBCDIC characters to ASCII characters. In addition, it converts the EBCDIC line feed character (X'15') to X'0D0A' which is the ASCII carriage return and line feed characters. In the last example, the **asc2eb** REXX program converts the ASCII characters to EBCDIC characters. In addition, it converts the ASCII carriage return and line feed character combination (X'0D0A') to X'15' which is the EBCDIC line feed character. It also eliminates the ASCII end of file character (X'1A').

Using Data Stored by OS/2 DFS Client Machines from OS/390

Data that was created by an application running on OS/2 will generally be in OS/2, or ASCII, format. When data is stored in an HFS file with a file system that is exported by DFS, the data is made available to OS/390 applications. In general, applications running on OS/390 do not expect ASCII format data. Generally, applications running on OS/390 expect EBCDIC format data.

You can use the **dd** command, the **tr** command and the **awk** command (available on the OS/390 shell) to convert data from one character representation to another. For syntax on the **dd** command, see the *OS/390 UNIX System Services Command Reference*, SC28-1892.

For example, the following command displays an ASCII file (**asciidat** stored by OS/2) on OS/390:

```
$ cat asciidat | tr -d '\015' | tr -d '\032' | dd conv=ebcdic | tr '\045' '\025'
```

ASCII format data can be stored in an HFS file in EBCDIC format. To store the contents of the ASCII format file **asciidat** in the file **hfsdata**, enter:

```
$ cat asciidat | tr -d '\015' | tr -d '\032' | dd conv=ebcdic | tr '\045' '\025' >hfsdata
```

The EBCDIC format file can be modified and stored in ASCII format. If **hfsdata** is modified, it can be stored back into the **asciidat** file with:

```
$ cat hfsdata | dd conv=ascii | awk '{ gsub("\205","\015\012");printf("%s",$0) }' >asciidat
```

Note: In the first two examples, the **tr** command deletes the ASCII carriage return character (X'0D') (\015 in octal) and the ASCII end of file character (X'1A') (\032 in octal). The **dd** command converts the ASCII characters to EBCDIC characters. In addition, it converts the ASCII line feed character (X'0A') to X'25' (\045 in octal). The **tr** command changes X'25' to X'15' (\025 in octal) which is the EBCDIC line feed character. In the last example, the **dd** command converts the EBCDIC characters to ASCII characters. In addition, it converts the EBCDIC line feed character (X'15') to X'85' (\205 in octal). The **awk** command changes X'85' to X'0D0A' (\015\012 in octal) which is the ASCII carriage return and line feed character combination.

Example REXX programs for OS/2 character conversion

EB2ASC.CMD

```
/* This is a REXX program that converts EBCDIC standard in (STDIN) text */
/*    to the equivalent ASCII standard out (STDOUT) text               */

/* Set up EBCDIC to (PC) ASCII translate tables. */
asc0= ,
 /* 0 1 2 3 4 5 6 7 8 9 A B C D E F */,
  '00010203CF09D37FD4D5C30B0C0D0E0F'X  ||  /* NUL SOH STX ETX 207 HT 211 DEL 212 213 195 VT FF CR SO SI */,
  '10111213C7B408C91819CCCD831DD21F'X;     /* DLE DC1 DC2 019 199 180 BS 201 CAN EM 204 205 131 029 210 031 */
asc20= ,
  '81821C84860A171B89919295A2050607'X  ||  /* 129 130 FS 132 134 LF ETB ESC 137 145 146 149 162 ENQ ACK BEL */,
  'E0EE16E5D01EEA048AF6C6C21415C11A'X;     /* 224 238 SYN 229 208 RS 234 EOT 138 246 198 194 DC4 NAK 193 SUB */
asc40= ,
  '20A6E180EB909FE2AB8B9B2E3C282B7C'X  ||  /* SPC 166 225 128 235 144 159 226 171 139 155 . < ( + | */,
  '26A9AA9CDBA599E3A89E21242A293BAA'X;     /* & 169 170 156 219 165 153 227 168 158 ! $ * ) ; ¬(circ) */
asc60= ,
  '2D2FDFDC9ADDDE989DACBA2C255F3E3F'X  ||  /* - / 223 220 154 221 222 152 157 172 186 , % _ > ? */,
  'D78894B0B1B2FCD6FB603A2340273D22'X;     /* 215 136 148 176 177 178 252 214 251 Oquote : # ' = " */
asc80= ,
  'F8616263646566676869996A4F3AFAEC5'X  ||  /* deg abcdefghi 150 164 243 175 174 197 */,
  '8C6A6B6C6D6E6F7071729787CE93F1FE'X;     /* 140 jklmnopqr 151 135 206 147 241 254 */
ascA0= ,
  'C87E737475767778797AEFC0DA5BF2F9'X  ||  /* 200 126 stuvwxyz 239 192 218 ﬄ >=(242) bul(249) */,
  'B5B6FDB7B8B9E6BBBCBD8DD9BF5DD8C4'X;     /* 181 182 253 183 184 185 230 187 188 189 141 217 191 “ 216 196 */
ascC0= ,
  '7B414243444546474849CBCABEE8ECED'X  ||  /* lcb ABCDEFGHI 203 202 190 232 236 237 */,
  '7D4A4B4C4D4E4F505152A1ADF5F4A38F'X;     /* rcb JKLMNOPQR 161 173 245 244 163 143 */
ascE0= ,
  '5CE7535455565758595AA0858EE9E4D1'X  ||  /* Rev-sl 231 STUVWXYZ 160 133 142 233 228 209 */,
  '30313233343536373839B3F7F0FAA7FF'X;     /* 0123456789 179 247 240 250 167 255 */
ascii=asc0||asc20||asc40||asc60||asc80||ascA0||ascC0||ascE0;

ebcdiclf = x2c('15')         /* EBCDIC line feed (LF)                 */

asciicr = x2c('0D')          /* ASCII carriage return (CR)            */
asciilf = x2c('0A')          /* ASCII line feed (LF)                  */

/* Loop through chars in input stream converting them to ASCII and    */
/* writing them to the output stream                                  */

/* For example,                                                       */
/*                                                                    */
/*  EBCDIC Input Stream             Resulting ASCII Output Stream    */
/*                                                                    */
/*      A   B                             A     B                    */
/*    X'C115C215'                      X'410D0A420D0A'               */
/*                                                                    */

x = charin()                 /* get the first char from input stream  */
do while chars() = 1         /* do while there is a char in x         */
    if x = ebcdiclf then     /*  if EBCDIC line feed (LF)             */
      do                     /*   convert to ASCII CR, LF             */
        g = charout(,asciicr)  /*  add ASCII CR                       */
        g = charout(,asciilf)  /*  add ASCII LF                       */
      end                    /*   end of convert to ASCII CR, LF      */
    else                     /*  if not EBCDIC line feed (LF)         */
      g = charout(,substr(ascii,1+c2d(x),1))/* convert EBCDIC to ASCII*/
    x = charin()             /*  get the next char from input stream  */
end                          /* all chars from input stream converted */
```

ASC2EB.CMD

```
/* This is a REXX program that converts an ASCII standard in (STDIN) text */
/*    to the equivalent EBCDIC standard out (STDOUT) text                 */

/* Set up (PC) ASCII to EBCDIC translate table */
ebc0= ,
 /* 0 1 2 3 4 5 6 7 8 9 A B C D E F */,
  '00010203372D2E2F1605250B0C0D0E0F'X  ||,  /* NUL SOH STX ETX EOT ENQ ACK BEL BS HT LF VT FF CR SO SI */
  '101112133C3D322618193F27221D351F'X;      /* DLE DC1 DC2 DC3 DC4 NAK SYN ETB CAN EM SUB ESC FS GS RS US */
ebc20= ,
  '405A7F7B5B6C507D4D5D5C4E6B604B61'X  ||,  /* spc !"#$%&'()*+,-./  */
  'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'X;      /* 0123456789:;<=>?  */
ebc40= ,
  '7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'X  ||,  /* @ABCDEFGHIJKLMNO  */
  'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D'X;      /* PQRSTUVWXYZﬄ\“_  */
ebc60= ,
  '79818283848586878889919293949596'X  ||,  /*  abcdefghijklmno  */
  '979899A2A3A4A5A6A7A8A9C04FD0A107'X;      /* pqrstuvwxyz       */
ebc80= ,
  '4320211C23EB249B7128384990BAECDF'X  ||,
  '45292A9D722B8A9A6756644A53685946'X;
ebcA0= ,
  'EADA2CDE8B5541FE58515F4869DB8E8D'X  ||,
  '737475FA15B0B1B3B4B56AB7B8B9CCBC'X;
ebcC0= ,
  'AB3E3B0ABF8F3A14A017CBCA1A1B9C04'X  ||,
  '34EF1E0608097770BEBBAC5463656662'X;
ebcE0= ,
  '30424757EE33B6E1CDED3644CECF31AA'X  ||,
  'FC9EAE8CDDDC39FB80AFFD7876B29FFF'X;
 /* 0 1 2 3 4 5 6 7 8 9 A B C D E F */,
ebcdic=ebc0||ebc20||ebc40||ebc60||ebc80||ebcA0||ebcC0||ebcE0;

asciicr = x2c('0D')          /* ASCII carriage return (CR)            */
asciilf = x2c('0A')          /* ASCII line feed (LF)                  */
asciief = x2c('1A')          /* ASCII end of file (EOF)               */

ebcdiclf = x2c('15')         /* EBCDIC line feed (LF)                 */

/* Loop through chars in input stream converting them to EBCDIC and   */
/* writing them to the output stream                                  */

/* For example,                                                       */
/*                                                                    */
/*  ASCII Input Stream              Resulting EBCDIC Output Stream    */
/*                                                                    */
/*      A     B                            A   B                      */
/*    X'410D0A420D0A1A'                 X'C115C215'                   */
/*                                                                    */

x = charin()                 /* get the first char from input stream  */
do while chars() = 1         /* do while there is a char in x         */
   if (x = asciicr | x = asciief) then /* if ASCII CR or EOF          */
     nop                     /*             skip it                   */
   else if x = asciilf then  /*            if ASCII LF                */
          g = charout(,ebcdiclf) /*          add EBCDIC LF            */
        else                 /*            if not ASCII LF, CR or EOF */
          g = charout(,substr(ebcdic,1+c2d(x),1)) /* ASCII to EBCDIC  */
   x = charin()              /*   get the next char from input stream */
end                          /* all chars from input stream converted */
```

Appendix D. DFS/NFS Secure Gateway

The DFS/NFS Secure Gateway provides a way for NFS clients to go to an NFS server that is exporting the DCE namespace and thereby access DFS data. This works on any machines that have the proper clients and servers. The NFS server machine must be exporting **/.../***cellname***/fs** and the NFS client machine must have mounted **/.../***cellname***/fs** on a local directory. For example (assuming that the cellname of the DFS data to be exported is **abc.com**), the following statement could be added to the OS/390 NFS Server exports data set:

```
/hfs/.../abc.com/fs
```

The following commands could be issued by the NFS client machine administrator to mount the DFS namespace (**/.../abc.com/fs**) that is exported by gateway machine **nfsgw1**.

```
# mkdir /dfs
# mount nfsgw1:/.../abc.com/fs  /dfs
```

However, even though the UNIX identity (uid) of the user flows from the NFS client machine to the NFS server, the DCE identity (uuid) that flows from the DFS client to the DFS Server is unauthenticated. So the NFS client can really only see "public" DFS data (see Figure 17).

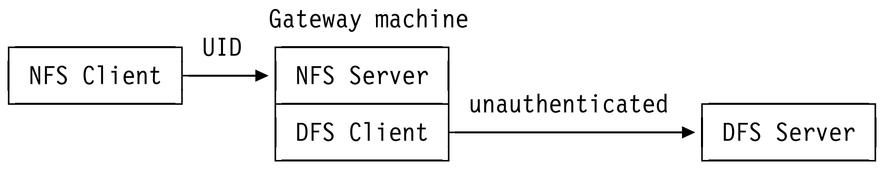


Figure 17. Unauthenticated Access to DFS Data

The NFS to DFS Authenticating Gateway for AIX adds the ability to map an NFS client user identity (including the IP address of the client machine) to a DCE identity that is unique to the NFS client user. This mapping (remote UNIX user to DCE identity) does not require the remote UNIX user to be enrolled on the gateway machine and uses the remote UID (and IP address) to map the NFS client user to the correct DCE identity (see Figure 18).

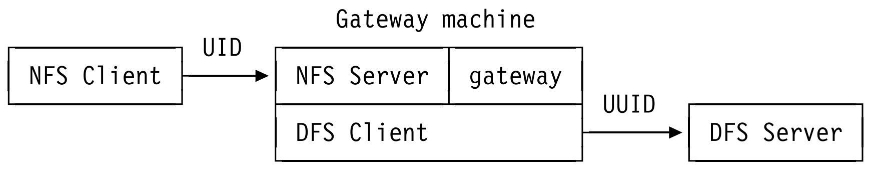


Figure 18. Authenticated Access to DFS Data

However, when an NFS client user wants to access OS/390 data through the OS/390 NFS server, the NFS client user must log on to OS/390 by the **mvslogin** command (issued on the NFS client machine) specifying an OS/390 user ID and password. This user must be enrolled on the OS/390 machine. The UID that is sent to the DFS client comes from the OS/390 security manager (e.g., RACF), not from the remote NFS client machine.

In addition, the OS/390 user ID must be automatically logged in with the DCE single sign-on facility from the DFS client. RACF has the capability to store an OS/390 user's DCE principal and password for DCE single sign-on support (see “DCE Single Sign-On from the DFS client” on page 30 for more information).

These two facilities (**mvslogin** and DCE single sign-on from the DFS client) effectively make a OS/390 NFS server that exports the DCE namespace a secure gateway. That is, no new **dfs** commands (e.g., dfsiauth) are required for this function.

Appendix E. Using Both SMB and DCE DFS

This appendix contains information that must be considered if you plan to use the OS/390 Distributed File Service DCE DFS protocols (enabled using the **_IOE_PROTOCOL_RPC=ON** environment variable) along with the SMB protocols (enables using the **_IOE_PROTOCOL_SMB=ON** environment variable).

Fileset IDs in dfstab

The **dfstab** and **devtab** files specify HFS file systems that are being exported. Those files are used for the SMB protocol and the DCE DFS protocol.

When you use the DFS protocol, filest IDs must be assigned by the **flserver**. (This is required whether SMB protocols are being used or not.) For an HFS fileset, a fileset ID is assigned by the **flserver** by using the **fts crfldbentry** command. The administrator enters the assigned fileset ID in the fileset's **dfstab** entry.

If you have been using DCE DFS protocols and you are now adding SMB protocols, the fileset IDs for any exported HFS filesets have already been assigned by the **flserver** and are already specified in the protocol. You can then create your **smbtab** entries to specify which shared directories should be available to PC clients. If you need to export any additional HFS filesets, the fileset IDs must be assigned by the **flserver** (even if they are only going to be accessed by SMB protocols).

Or, if you have been using SMB protocols and you are now adding DCE DFS protocols, the HFS fileset IDs specified in the **dfstab** must be changed to numbers that have been assigned by the **flserver**. This means that you must enable and start the Distributed File Service Server for DCE DFS protocols, assign the HFS fileset IDs by using the **fts crfldbentry** command, update your **dfstab** entries for the HFS fileset IDs, and then export the filesets (use the **modify dfs,start export** system command of the **dfsexport** command). If you need to add any HFS filesets, their fileset IDs must also be assigned by the **flserver**

Appendix F. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This *Distributed File Service Administration Guide and Reference* primarily documents information that is NOT intended to be used as Programming Interfaces of Distributed File Service.

This *Distributed File Service Administration Guide and Reference* also documents an intended Programming Interface that allows the customer to write programs to obtain the services of Distributed File Service. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

End of Programming Interface information

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

AIX	AIX/6000	BookManager
DFSMS	DFSMS/MVS	DFSMShsm
IBM	IBMLink	Language Environment
Library Reader	MVS/ESA	OpenEdition
OS/2	OS/390	RACF
RS/6000		

DFS is a trademark of Transarc Corporation in the United States, or other countries, or both.

Bibliography

This section lists and provides a brief description of each publication in the OS/390 Distributed File Service library. Also listed are publications from the OS/390 DCE library that may be useful.

OS/390 Distributed File Service Publications

This section lists and provides a brief description of each publication in the OS/390 Distributed File Service (DFS) library.

Administration

- *OS/390 Distributed File Service DFS Configuring and Getting Started*, SC28-1722

  This book helps system and network administrators configure the OS/390 Distributed File Service.

- *OS/390 Distributed File Service Administration Guide and Reference*, SC28-1720

  This book introduces the DFS concepts to system and network administrators and provides an in-depth understanding of DFS, its uses and benefits. This book also provides reference information for the commands and files used by system and network administrators to work with DFS.

- *OS/390 Distributed File Service SMB Administration Guide and Reference*, SC24-5882

  This books provides guidance and reference information for system and network administrators to use when they work with the Server Message Block (SMB) support of the IBM OS/390 Distributed File Service base element of OS/390. SMB is a protocol for remote file/print access used by Windows.

Reference

- *OS/390 Distributed File Service DFS Messages and Codes*, SC28-1724

  This book provides detailed explanations and recovery actions for the messages, status codes, and exception codes issued by the OS/390 DFS.

OS/390 DCE Publications

This section lists and provides a brief description of each publication in the OS/390 DCE library.

Overview

- *Distributed Computing Environment: Understanding the Concepts*, GC09-1478

  This book introduces Open Software Foundation (OSF) DCE. It describes the technology components of DCE, from a high-level overview to a discussion of the interdependencies among the components.

- *OS/390 DCE Introduction*, GC28-1581

  This book introduces OS/390 DCE. Whether you are a system manager, technical planner, OS/390 system programmer, or application programmer, it will help you understand DCE, and evaluate the uses and benefits of including OS/390 DCE as part of your information processing environment.

Planning

- *OS/390 DCE Planning*, SC28-1582

  This book helps you plan for the organization and installation of OS/390 DCE. It discusses the benefits of distributed computing in general, and describes how to develop plans for a distributed system in an OS/390 DCE environment.

Administration

- *OS/390 DCE Configuring and Getting Started*, SC28-1583

  This book helps system and network administrators configure OS/390 DCE.

- *OS/390 DCE Administration Guide*, SC28-1584

  This book helps system and network administrators understand OS/390 DCE, and tells how to administer it from the batch, TSO, and shell environments.

- *OS/390 DCE Command Reference*, SC28-1585

  This book provides reference information for the commands that system and network administrators use to work with OS/390 DCE.

- *OS/390 OpenEdition DCE User's Guide*, SC28-1586

  This book describes how to use OS/390 DCE to work with your user account, use the directory service, work with namespaces, and change access to objects that you own.

Application Development

- *OS/390 DCE Application Development Guide Introduction and Style*, SC28-1587

  This book assists you in designing, writing, compiling, linking, and running distributed applications in OS/390 DCE.

- *OS/390 DCE Application Development Guide Core Components*, SC28-1588

  This book assists programmers in developing applications using application facilities, threads, remote procedure calls, distributed time service, and security service.

- *OS/390 DCE Application Development Guide Directory Services*, SC28-1589

  This book describes the OS/390 DCE directory service and assists programmers in developing applications for the cell directory service and the global directory service.

- *OS/390 DCE Application Development Reference*, SC28-1590

  This book explains the DCE Application Program Interfaces (APIs) that you can use to write distributed applications on OS/390 DCE.

Reference

- *OS/390 DCE Messages and Codes*, SC28-1591

  This book provides detailed explanations and recovery actions for the messages, status codes, and exception codes issued by OS/390 DCE.

Security Server Publications

This section lists and provides a brief description of books in the Security Server library that may be needed for the DCE Security Server and for RACF interoperability.

- *OS/390 OpenEdition DCE Security Server Overview*, GC28-1938

  This book describes the DCE security server and provides a road map for DCE security server information in the OS/390 DCE library.

- *OS/390 Security Server (RACF) Security Administrator's Guide*, SC28-1915.

  This book explains RACF concepts and describes how to plan for and implement RACF.

Index

Readers' Comments

OS/390®
Distributed File Service DFS
Administration Guide and Reference

Publication No. SC28-1720-06

You may use this form to report errors, to suggest improvements, or to express your opinion on the appearance, organization, or completeness of this book.

Date: ___________________________________

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note

Report system problems to your IBM representative or the IBM branch office serving you.
U.S. customers can order publications by calling the IBM Software Manufacturing Solutions at **1-800-879-2755**.

In addition to using this postage-paid form, you may send your comments by:

FAX	1-607-752-2327	Internet	pubrcf@vnet.ibm.com
IBM Mail	USIB2L8Z@IBMMAIL	IBMLink	GDLVME(PUBRCF)

Would you like a reply? ___**YES** ___ **NO** If yes, please tell us the type of response you prefer.

___ Electronic address: ____

___ FAX number: ___________________________________

___ Mail: (Please fill in your name and address below.)

Name	Address
Company or Organization	
Phone No.	

Readers' Comments
SC28-1720-06

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Department G60
International Business Machines Corporation
Information Development
1701 North Street
ENDICOTT NY 13760-5553

Fold and Tape **Please do not staple** Fold and Tape

SC28-1720-06

Cut or Fold
Along Line

Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC28-1720-06

Spine information:

IBM OS/390 DFS Administration Guide and Reference