The Distributed File System (DFS) for AIX/6000

Document Number GG24-4255-00

May 1994

International Technical Support Center
Austin

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

First Edition (May 1994)

This edition applies to:

- AIX DCE Enhanced Distributed File System/6000 (5765-121)
- AIX DCE Base Services/6000 (5765-117)
- AIX DCE Cell Directory Server/6000 (5765-119)
- AIX DCE Security Server/6000 (5765-118)

for use with the AIX 3.2.5 Operating System.

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

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 948S, Building 821 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Abstract

This document describes the features contained in the AIX DCE Enhanced Distributed File System/6000 Version 1.1 and the AIX DCE Base Services/6000 Version 1.2 software products. It provides useful discussion and examples of practical features of this product. We have put a fair amount of examples in this book to explain the manner of the use of the large DFS administration command set. The purpose is to give some guidance to the technical network administrators as well as to decision makers considering the implementation of the Distributed File System (DFS).

Although this document was written mainly for technical system and network administrators, anyone can benefit by picking up the major concepts that are introduced.

(239 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help people who are using or intending to use the:

- AIX DCE Base Services/6000 Version 1.2 and
- AIX DCE Enhanced Distributed File System/6000 Version 1.1 products.

It is our intent to show the capabilities of these products in the area of the Distributed File System (DFS). We hope to clarify how to use these products as well as to describe the various functions that these products provide. This book is written for both the administrators of these products as well as anyone who needs a detailed overview as to the function of the DFS. The information in this publication is not intended as the specification of any programming interfaces that are provided by the AIX DCE Base Services/6000 Version 1.2 and AIX DCE Enhanced Distributed File System/6000 Version 1.1 products. See the PUBLICATIONS section of the IBM Programming Announcement for the AIX DCE Base Services/6000 Version 1.2 and AIX DCE Enhanced Distributed File System/6000 Version 1.1 products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT USA 06904.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
AIXwindows	IBM
InfoExplorer	OS/2
RISC System/6000	

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

Open Software Foundation	The Open Software Foundation
DCE	The Open Software Foundation
Episode	Transarc Corporation
Transarc	Transarc Corporation
DIGITAL	Digital Equipment Corporation
HP/UX	Hewlett-Packard Company
OSF	Open Software Foundation, Inc.
NFS (Network File System)	SUN Microsystems, Inc.
UNIX	Unix System Laboratories, Inc.
X-Windows	Massachusetts Institute of Technology
Motif	Open Software Foundation, Inc.
DEC	Digital Equipment Corporation
ULTRIX	Digital Equipment Corporation

Preface

This document is intended to assist people that are planning a Distributed File System installation or people who have already installed DFS on their machines. This book contains our experiences of working with the DFS. We have documented many examples and hints that we think will be useful to both administrators who are about to install the DFS on their machines as well as those who have already done so. This book also contains an overview of the DFS function from a practical system administrator point of view.

This document is intended for system/network administrators, planners, users or anyone who needs to know the capabilities/limitations of the DFS.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “DFS Introduction”

  This chapter describes the Distributed File System (DFS) of the Distributed Computing Environment (DCE). The capabilities as well as its basic operation are included here. Some DFS terms are also described.

- Chapter 2, “DFS Concepts and File System Interaction”

  This chapter describes the various functions that machines can perform in the DFS. This chapter also discusses the interaction of the DFS with other types of filesystems.

- Chapter 3, “DFS Data Storage and Manipulation”

  This chapter describes various concepts of DFS such as filesets, tokens, mount points and replication. A discussion of backup policies is also presented. The purpose is to give an overview into some of the complexities of the various components of the DFS product.

- Chapter 4, “Installing and Configuring DFS”

  This chapter provides an overview of the AIX/6000 DCE product family as well as the DFS members of that family. A complete example of how to install the DFS and to do some simple administration is given. From this chapter, an administrator can configure a working DFS environment.

- Chapter 5, “DFS Administration Tasks”

  This chapter discusses the administration tasks that are associated with the DFS. These tasks include fileset management, monitoring and controlling the DFS file exporters, backup/restore concepts as well as cache management and security management. Many examples are given so that the functions performed by these actions can be clearly understood.

- Chapter 6, “Security”

  This chapter describes the security aspects that are involved when working with the DFS. The concept of Access Control Lists (ACLs) are discussed and examples are given for both a single and multicell environment. The ACLs can be one of the most important (and most confusing) aspects of DFS.

- Chapter 7, “Some Real Life Examples with DFS”

This chapter presents some examples of some of the things that you can use the DFS to accomplish for you. We have used this chapter to describe a common installations image distributor, a common documentation distributor and a common binary file distributor.

- Chapter 8, “DFS Troubleshooting”

  This chapter describes some guidelines that may be used in troubleshooting the DFS.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *Understanding DCE Concepts*, GC09-1478
- *AIX/OSF DCE Release Notes*, GC23-2434
- *Developing DCE Applications for AIX and OS/2*, GG24-4090
- *OSF DCE for AIX, OS/2 and DOS*, GG24-4144
- *AIX DCE Administration Guide*, SC23-2475
- *AIX DCE Overview*, SC23-2477
- *AIX DCE Messages Reference*, SC23-2583
- *OSF/DCE Introduction To DCE (PRENTICE HALL)*, SR28-4991
- *OSF/DCE User's Guide and Reference (PRENTICE HALL)*,SR28-4992
- *OSF/DCE Administration Reference (PRENTICE HALL)*, SR28-4993
- *OSF/DCE Application Development Guide (PRENTICE HALL)*, SR28-4994
- *OSF/DCE Application Development Reference (PRENTICE HALL)*, SR28-4995

International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Centers Technical Bulletins, GG24-3070.

Acknowledgments

The advisor for this project was:

Ron Smetana
International Technical Support Organization, Austin

The authors of this document are:

Ron Smetana
IBM Austin

Clarice Kobayashi
IVIX Sistemas Abertos Brazil

Peter Koepp
IBM Germany

Joao Takeda
IBM Brazil

Soo Keun Na
IBM Korea

The reviewers for this book included:

Michael Mueller
IBM Germany

Tony Rynan
IBM Australia

Craig Powell
IBM United Kingdom

This publication is the result of a residency conducted at the International Technical Support Organization, Austin.

We would like to offer our thanks to the IBM-Austin DCE Development team for their hard work, invaluable advice and assistance in the reviewing of this publication.

This book is dedicated to those IBMers that have worked hard throughout their career and now have time to read this book.

Chapter 1. DFS Introduction

Distributed File System (DFS) technology provides the ability to access and store data at remote sites similar to the techniques used with NFS** (Network File System). It extends the view of a local, and therefore limited in size, file system to a distributed file system of almost unlimited size located on several remote systems. A distributed file system has many advantages over a centralized system. These advantages include providing access to files from anywhere in the world, higher availability through replication and providing users on systems the ability to access data from a nearly unlimited data space.

The Open Software Foundation** (OSF**) considers distributed file systems an essential part in a distributed computing environment. For that reason, OSF solicited DFS technology for inclusion in its DCE** technology. The DFS technology was submitted to OSF by Transarc. Transarc is a company that had been selling a distributed filesystem called the Andrew File System (AFS). DFS is basically an enhanced version of the AFS technology that has been integrated in with the base DCE technology.

1.1 DFS File System Structure

The distributed file system from DCE Distributed File Services (DFS) is a collection of several file systems located on distributed systems. All these file systems are mounted into a single virtual file system space with a single namespace. The end user has direct access to all files in this distributed file system without knowing where the physical files reside.

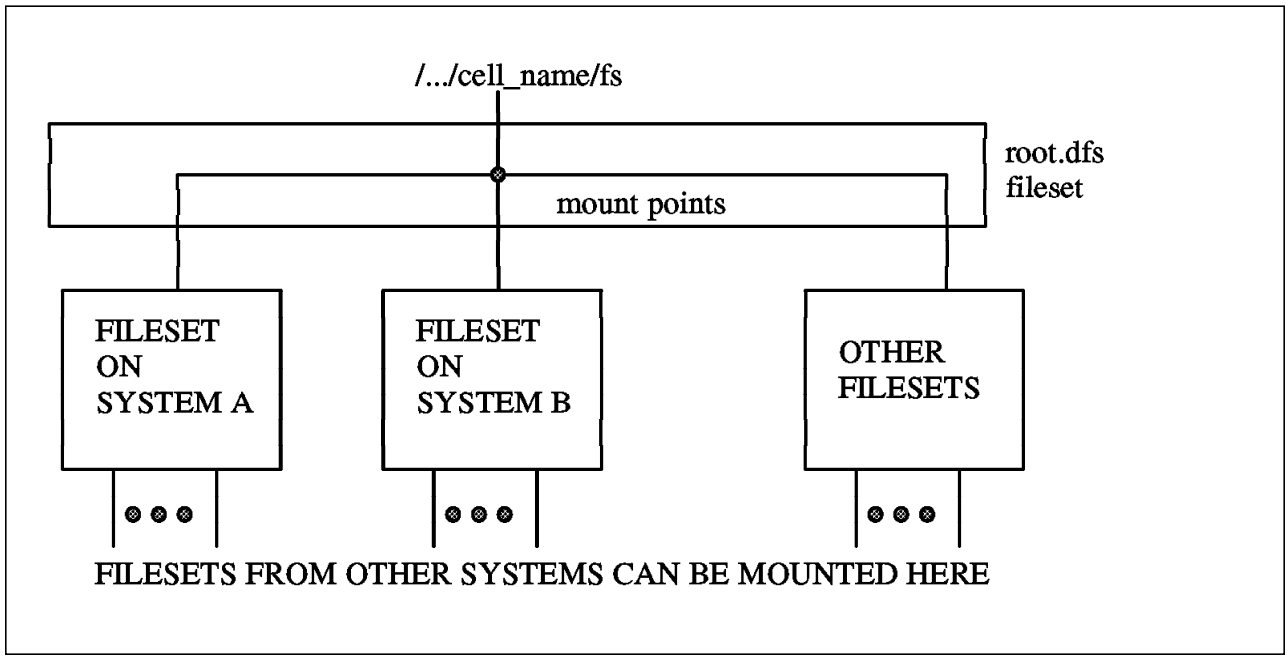


Figure 1. Single Image View of the DFS Namespace

Figure 1 shows that a DFS file system has a hierarchical structure consisting of directories and files as it is known from other UNIX** systems. The root of the DFS file structure is a junction in the DCE naming space called /.../<cell_name>/fs. There is a short method to write to the root fileset of the

cell that you are logged into. This is "/.:/fs" or just "/:". Both of these are equivalent.

A fileset is similar to a UNIX (or AIX*) file system. It is a hierarchical collection of directories and files. These filesets which can reside on many different servers can be mounted into the DFS namespace.

1.2 DFS Operation

DFS is built on the concept of a client-server architecture. The server provides data and the client uses the data. The communication between the server and the client is handled with DCE remote procedure calls. Those systems in a DFS environment that own file systems export them and users on other systems access these file systems. We call the file exporting machine the DFS File Server System and the importing machines the DFS client systems. A machine can also be both a server and a client. Figure 2 shows the client-server nature of DFS.

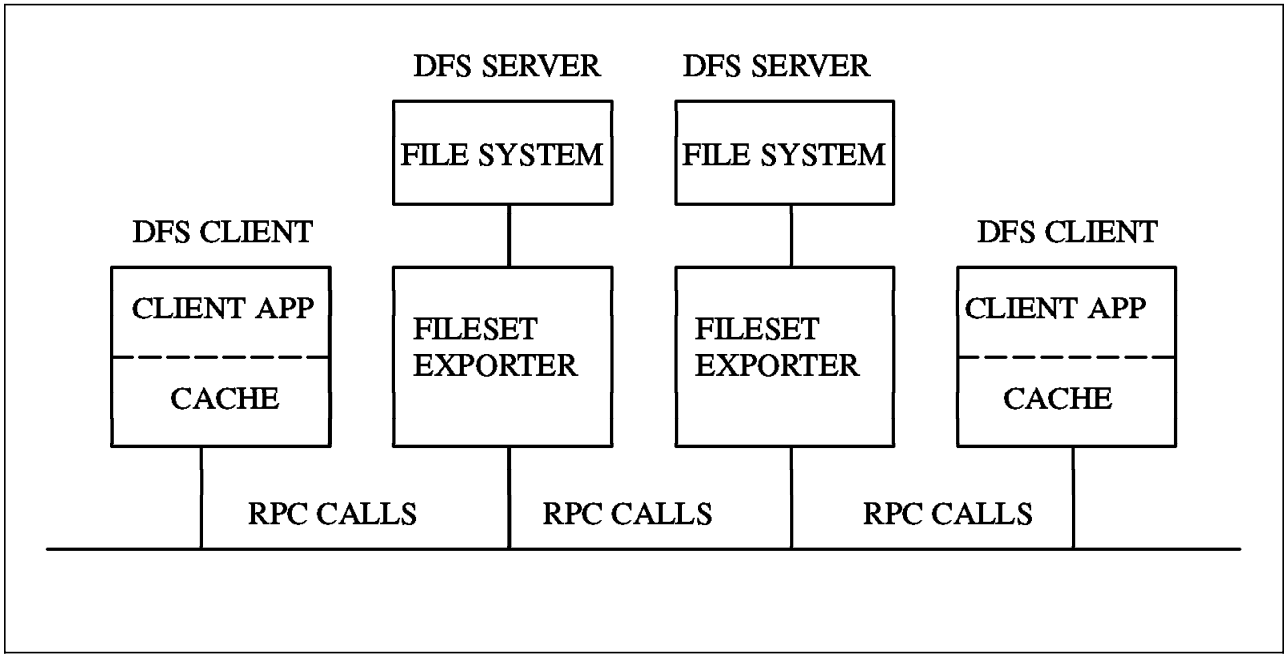


Figure 2. DFS Client and DFS Server File Server Operations

Each DFS server system runs a Fileset Exporter which makes file systems available to the DFS filespace. DCE cells may have one or more DFS File Servers. As mentioned previously, a system can function as both a DFS client and DFS server. DFS clients run the Cache Manager, which caches data from the File Exporter in memory or on a local disk. The Cache Manager is an important component of DFS to improve performance and availability.

All DFS server systems which want to export their filesets must register their exported filesets at a system which is called the Fileset Location Server. The Fileset Location Server (sometimes called the Fileset Location DataBase) (FLDB) maintains a database of all filesets. This database is used to keep track of the physical locations where filesets are stored. If a DFS client requires access to one of the filesets, it sends it's request first to the FLDB to inquire information about where this fileset is physically located. After receiving location information,

it then contacts the actual DFS File Server. Figure 3 on page 3 shows this process. Subsequent access can then go directly to the DFS File Server.

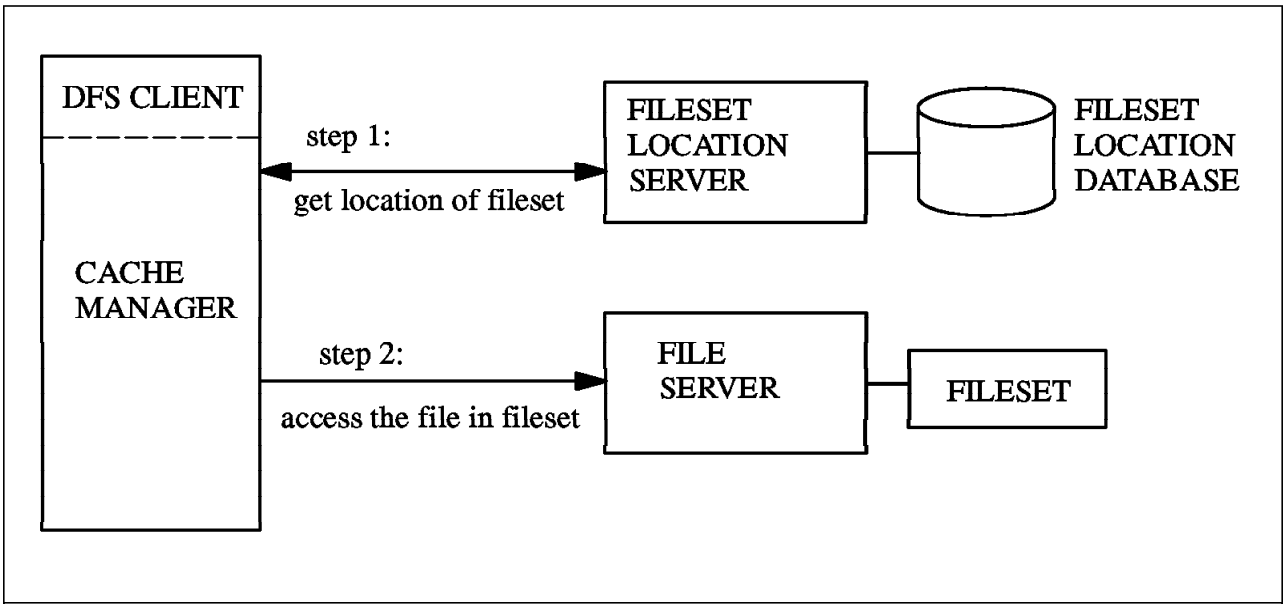


Figure 3. How a DFS Client Finds a DFS Server

1.3 DFS Machine Roles

We have already discussed the basic function of DFS file server machines, the DFS client machines and the Fileset Location Server. There are other functions or services that may be provided by one or more machines. These services can be categorized into machines roles. Figure 4 on page 5 shows the possible types of machine roles that may exist in a DFS environment. Not all of the machine roles are required. A machine can actually have more than one role. Actually, all of these machine roles may be combined in a single system. More than likely, they will be spread throughout the cell. From this figure, you can see that there are eight different machine roles. They can be described as follows:

File Server Machine: A File Server Machine runs the basic set of processes necessary for exporting DFS filesets to be accessible to DFS client machines. The administration for this machine is done by an appointed set of administrators that do not necessarily have to be logged into this file server machine.

Private File Server Machine: A Private File Server machine is very similar to a general File Server Machine with the exception that it controls its own administration and is therefore independent from a System Control Machine. Therefore a user on this machine can export file systems and have full control over the filesets that he has exported.

Fileset Location Server: This is also known as the Fileset Location Database Machine. This machine maintains the Fileset Location Database (FLDB) which holds information about exported Filesets.

DFS Clients: The DFS Client Machine allows a user (or multiple users) to access files from the DFS file servers. This machine can also be configured as a Private File Server machine to export file systems.

System Control Machine: The DFS environment can be set up to contain multiple regions that can be administered separately. These regions are called domains. There is a single machine as the System Control machine for a domain. This machine updates other machines in the DFS domain with identical versions of common configuration files such as administrative lists. An administrative list is a file containing a list of users or groups that have the proper authority to perform certain administrative functions on file servers and fileset location databases.

Binary Distribution Machine: A Binary Distribution Machine is used to distribute identical versions of DFS binary files to other machines in the cell. These binary distribution machines are used so that DFS servers can receive the proper binaries that relate to their type of CPU and version of operating system.

Backup Database Machine: The Backup Database Machine maintains a database which holds information filesets, fileset groups and the schedules that you want to back them up on. This database is used by the Backup System to provide regular dumps of the DFS filesets.

Tape Coordinator Machine: The Tape Coordinator Machine controls tape units which are physically attached. The Tape Coordinator Machine works in cooperation with the Backup System so that filesets can be backed up to specified tapes.

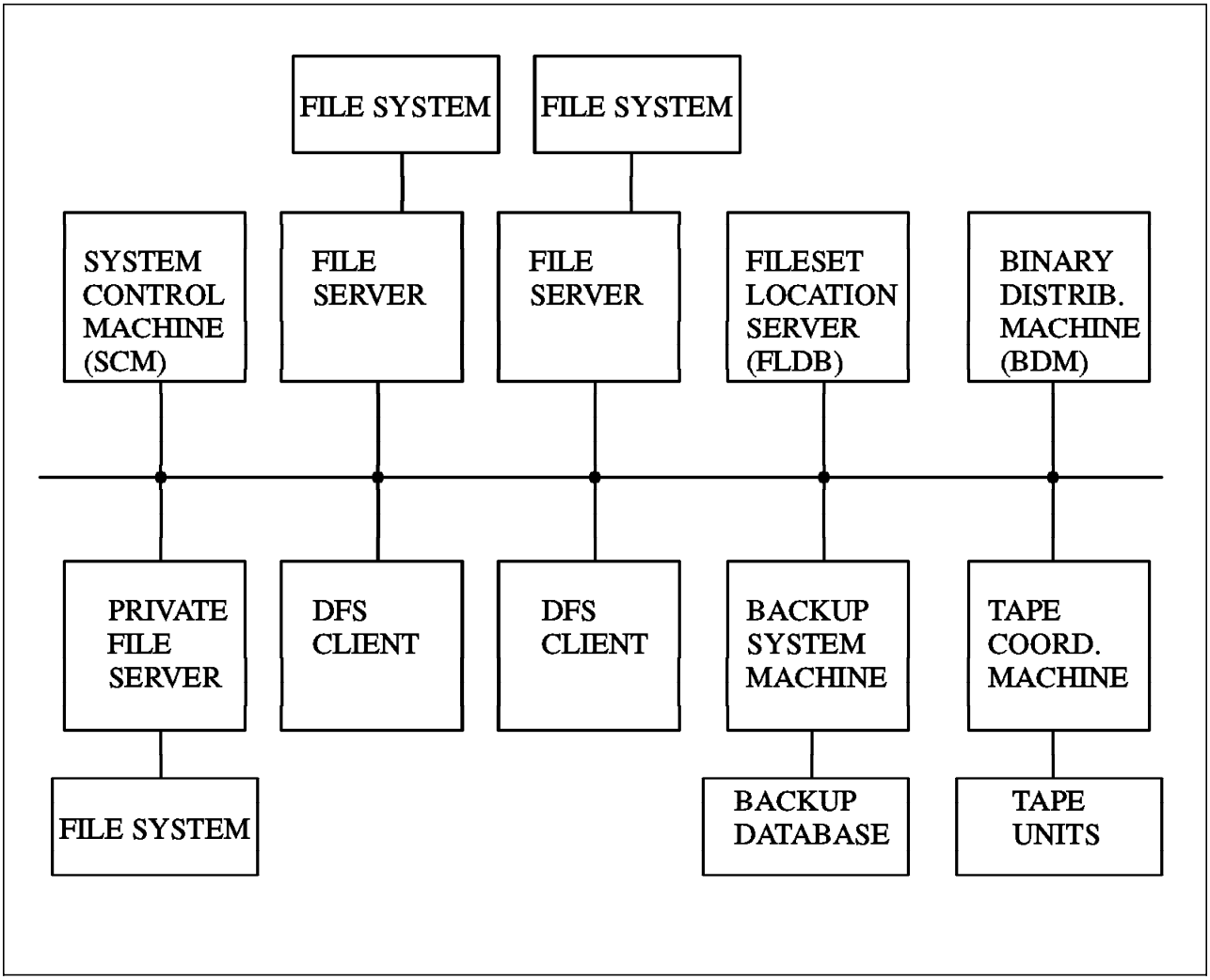


Figure 4. DFS Machine Roles

1.4 Basic Benefits of DFS

The DCE DFS provides many benefits over other types of distributed file systems. Some of these benefits are listed below.

1.4.1.1 Uniform Filespace

The DFS Filespace is uniform. This means that a particular file or directory can be accessed by using the same pathname from any DFS client. This is unlike other file systems that are mounted into a user's local file system at a particular mount point. The major benefit is that a user can login to any DFS client machine in the DCE cell and work with his files as though he was logged on to his own machine.

1.4.1.2 Caching on the DFS Client Machine

The DFS client machine administrator can define the size of the cache that is used for caching files in the DFS. The effect of caching a file is the same as a user having that file stored locally.

1.4.1.3 Finer Granularity for Access Control

A user is able to define exactly who can use his files and directories. Typical UNIX access control is limited to a particular user, a particular group and everyone else. By using the DCE Access Control Lists (ACLs) a user is able to define *which* user can have a particular type of access to a file or directory. The fact that the file owner can define the access permissions to his file for specific users gives the owner tight control over his files. The fact that DFS works with the underlying DCE security system guarantees that users cannot pretend to be other users in order to gain access to a file or directory.

1.4.1.4 DCE LFS Filesets Can Be Replicated (Future Release)

The DCE Local File System (LFS) has been architected to allow the replication of read-only filesets. This will allow users to store their files on more than one file server. The result is a file system with higher availability as a user will still be able to get to read-only data in the event that a file server housing read-only data becomes inoperable.

1.4.1.5 Ability to Establish Binary Distribution Machines

A DFS administrator will have the ability to set up machines that will download specific binary programs to machines of a specific architecture type as well as a particular version of operating system.

1.4.1.6 DFS Will Work on Other Vendor′s Platforms

DFS will be able to work with any vendor′s platforms that implement the DFS and the underlying DCE base. From the user point of view, this means that a DFS client may be working on a file that could be stored on a DFS file server anywhere in the DCE cell. This could be anything from a workstation to a mainframe. The user will not even know (or care) where his files are stored.

1.4.1.7 DFS Has Built-in Backup Capability

The DFS architecture defines a backup server and a tape controller machine role that is user to perform both full and incremental backups on DFS filesets. These backups may be scheduled to occur at specific times.

1.4.1.8 Diverse Administration Options

The DFS has administration options that can be set up to allow different administrators to administrate portions of the DFS namespace. Thus, you can define particular file servers and FLDBs to be administrated by particular users or groups.

1.5 DFS Relation to DCE

DFS is built on top of the underlaying DCE Services. It takes advantage of the lower level services of DCE such as RPC, Security Services, Directory Services, and Time Services. Before DFS can be configured on a machine it requires the following DCE components be installed, configured and running in the cell:

1. Security Server
2. Directory Server

3. DCE Time Servers (at least three are recommended)

An understanding of DCE (in particular the Security and Directory Services) is essential for an understanding of DFS. See Figure 5 for an example of a base DCE cell.

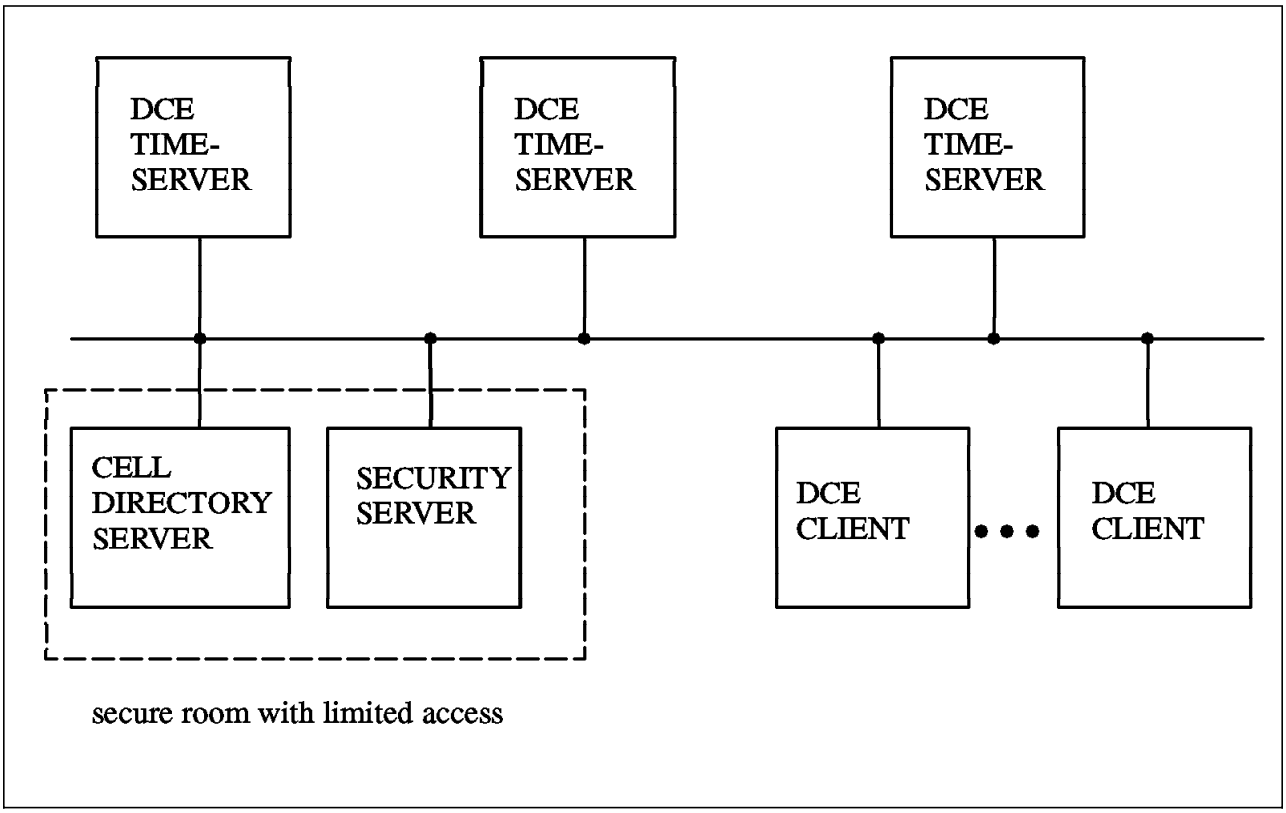


Figure 5. Basic DCE Cell Configuration

The Security Service provides mechanisms for authentication, privilege service and registry service. All of these services are used with DFS. A user of the DFS file system must login to DCE and therefore requires **authentication** by the DCE Security Service. The security server then returns authorization and privilege information associated with that user. This information is used to determine the type of access rights for DFS files and directories.

The DCE Cell Directory Service (CDS) manages objects contained in the cell. These objects can be things like DFS file servers, DCE timeservers or even hosts. The purpose of the CDS is to find the address of a particular server in the cell. Every DFS path name consists of a leading /.../<cell_name>/fs. The *fs* is called a junction point. The components up to and including the junction point must be passed to CDS for resolution. Components after the *fs* are resolved by the DFS file system. The junction fs acts like a boundary between the CDS namespace and the DFS filespace. Note that entries in the CDS namespace are manipulated with DCE commands such as cdscp while files and directories in the DFS filespace are manipulated with standard AIX file system commands.

The DCE Time Service (DTS) provides synchronization of the clocks on all systems in the DCE cell. With DFS it is important that DFS clients and DFS servers all have the same idea of what the correct time is. There are (at least) two reasons that having the same time is important. First, if all machines storing files on the file server were set to slightly different times, one would never be

sure about the accuracy of the file timestamp information. You would not be able to put events in their correct sequence. For example, you would not know what files were created in what order on a file system. The second reason is due to the security services. After you do a dce_login, you request permission to a DFS file server. The security server then gives you *credentials* that say what user you are and what groups you belong to. When you access a DFS file server, you give it the credential information that proves your true identity. This information will expire in some predefined time limit. Therefore the DFS file server and your client machine must have the same general idea of what time it is.

1.6 DFS Terminology

For the remainder of this publication you will be confronted with new terms. To give you a start to understand these terms we have summarized some of the important terms in this section.

File Servers and Exporters

FLDB FLDB stands for Fileset Location Database. This is a database in a DFS cell which holds location information about filesets. DFS clients contact the FLDB database to learn which DFS servers export which filesets that house the files and directories that they are interested in.

File Server A File Server machine runs the basic set of processes necessary for storing and exporting LFS fileset and non-LFS fileset (JFS) data.

File Exporter The File Exporter is a process called the fxd process. There is one File Exporter running on each File Server Machine.

Filesets and Aggregates

Aggregate A logical unit of disk storage similar to a disk partition. On AIX, a DCE LFS aggregate is a logical volume that has been formatted as a DCE LFS physical file system by the DFS `newaggr` command. A DCE LFS aggregate can contain multiple DCE LFS filesets. A standard UNIX file system (AIX JFS file system) exported into the DFS file space by a DFS File Server is referred to as aggregate or a non-LFS aggregate.

Fileset

A hierarchical grouping of files managed as a single unit; this is the basic unit of data administration in DFS. DCE LFS supports multiple filesets within a single aggregate. When a standard UNIX file system, for example AIX JFS, is used with DFS the entire file system is considered one fileset.

DCE LFS The log-based physical file system provided with DFS. The DCE LFS supports multiple filesets within a single aggregate, fileset replication, fast system restarts, and DCE access control lists.

JFS The Journaled File System (JFS) is the native file system with IBM AIX. A JFS file system can be used with DFS by exporting it from a DFS File Server as an aggregate and registering it as a fileset with the DFS Fileset Location Database server. In the context of DFS

documentation, JFS file systems are referred to as non-LFS or UFS aggregates and filesets.

Fileset Header The fileset header is stored on each DCE LFS fileset. It holds fileset information such as name, type of fileset, quotas and time of last update.

Quotas The amount of data in kilobytes (KB) that a user is allowed to store on an DCE LFS fileset is called a quota. When this quota limit is reached, an administrator can then take appropriate action.

Cache Management on DFS Clients

Cache The Cache is an area in memory or on disk of the DFS client machine that caches data from File Exporters. The location of the cache defaults to /opt/dcelocal/var/adm/dfs/cache where data is cached in files called *V-files*.

Cache Size The Cache Size is the total amount of disk or memory space provided for the Cache Manager to perform DFS caching. The default size for disk caching is 10 Megabytes but it can be increased.

Chunk Size The Chunk Size is the amount of data fetched and stored by the Cache Manager within a single operation. The default Chunk Size is 64 kilobytes. This can also be customized.

Cache Manager The Cache Manager fetches and caches files from File Exporters on behalf of applications running on DFS client machines. An example of an application running on a DFS client could be an editor.

DFS Backup System

Backup System With the DFS Backup System you can copy data from DCE LFS and non-LFS filesets to tape and restore this data. The DFS Backup System allows you to maintain full backups and/or incremental backups.

Backup Database The Backup Database is a database used by the Backup System of DFS. It holds information about schedules for backup as well as what filesets or groups of filesets that you want to backup.

Tape Coordinator Machine The Tape Coordinator Machine is the system which has one or more tape units physically attached and which operates together with the the DFS Backup System to physically write and read data to the tape.

Clone A Clone is a synonym for a backup of a fileset created with the `fts clone` command. The name clone has its origin in the fact that the backup does not actually copy data, but uses pointers to the same data blocks as the original file. This clone of a fileset can be thought of as a *snapshot* of the file system at a particular point in time. A DFS administrator can then back up this clone while a user continues to use the original fileset.

General Items

Domain Name Server (DNS) In TCP/IP, hierarchical naming is known as the domain naming system (DNS) and uses the DOMAIN protocol. A domain name server works across cells and resolves hierarchical naming structures to Internet addresses.

Administrative Domain A DFS administrative domain is a collection of machines in the same cell configured for administration as a single unit. This means that a particular DFS administrator has been given the proper permissions to administer all machines in this domain.

Administrative Lists DFS administrative lists are files that define the principals and groups that can perform administrative actions on DFS server machines. This administrative list pertains to the administrative domain.

Mount Point A DFS mount point is a specific type of symbolic link which acts as a reference from a directory to a fileset. When a DFS fileset is mounted on a mount point, DFS clients (users) with proper permissions have access to files and directories in this fileset.

Local Mount Point A DFS Fileset may also be mounted locally. Files in a locally mounted fileset have two paths, the DFS path and the local path. A local mount point could be used if you are logged in to the machine that is the file server and the network becomes inoperable. You would then locally mount the file system and access the data as though you were a regular AIX user.

1.7 Example of a DFS Cell

We have already seen a basic DCE cell configuration. We will now show what a basic DFS configuration would be. Please refer to figure Figure 6 on page 11. This figure not only shows a DCE configuration, but now different DFS components have also been added. The purpose is to show the different DFS machine roles. We will discuss planning issues in Chapter 2, “DFS Concepts and File System Interaction” on page 13.

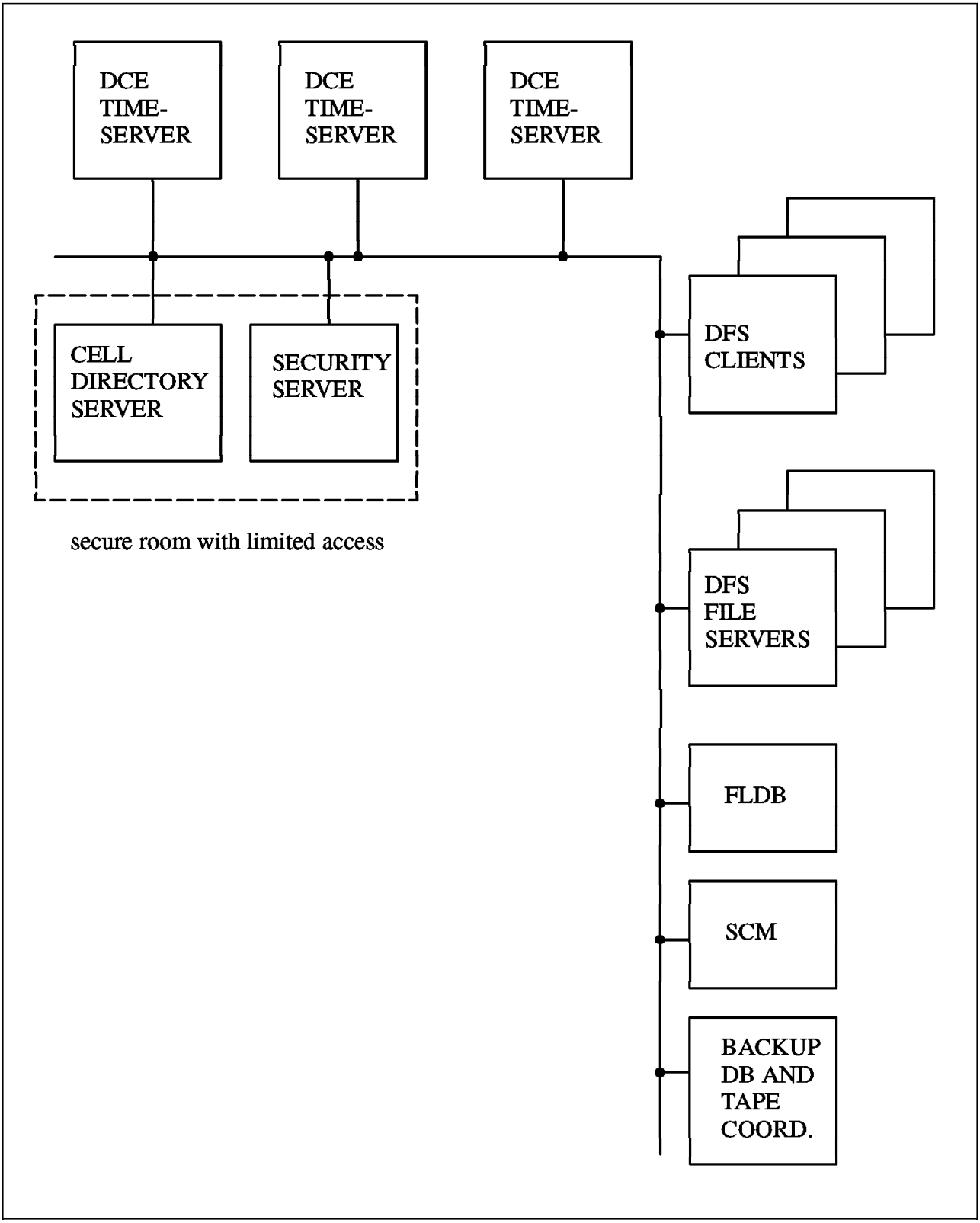


Figure 6. Fully Configured DFS Cell

Note that a DCE cell has certain requirements. Certain requirements are also necessary to operate the DFS. The DCE requirements that you must have are:

- At least one Cell Directory Server (CDS)

- One security server (IBM will support replicated security servers in a future release)
- At least three Distributed Time Servers (DTS)

The above requirements are DCE requirements and not DFS requirements. The DCE cell can operate with the above requirements. When you add the DFS capability to the DCE cell, here is what you must have:

- All DFS clients and servers must be minimally configured as DCE clients
- You must have at least one Fileset Location Database (FLDB) Machine
- You must have one System Control Machine (SCM) per administrative domain

Note that you can have as many DFS file servers and clients as you want. Also note that the backup Database machine as well as the tape coordinator are optional.

Figure 6 on page 11 shows that some of the machine roles such as the System Control Machine (SCM) and the FLDB and Backup Database machines are put on separate machines. This has just been done for clarity. In practice, more than one of these functions will often be implemented on just one machine.

Chapter 2. DFS Concepts and File System Interaction

This chapter will detail the various machine roles that are involved in a DFS configuration. As mentioned in Chapter 1, DFS provides for certain DFS tasks to be associated with different machines. We will describe these tasks and how they relate to administering a DFS environment called a domain. We will also show how DFS works with other types of file systems that are found in a UNIX or AIX environment. In particular, we will show the relationship of DFS to the AFS file system and to the Network File System (NFS).

2.1 DFS Model

Distributed systems can provide many advantages over centralized systems. These benefits can include higher availability of data and resources, the ability to share information throughout a large network and the ability to manage information and make it available as if it were local. Distributed systems can also take advantage of a particular machine's hardware. Another apparent benefit of distributed systems is that they can be incrementally scaled as user need for these type of machines increase.

DFS joins the file systems from different machines into a single global file system (filespace) accessible to many machines. Working with DFS is similar to working with the file system of our local machine.

The DFS model is considered distributed because data stored on many different DFS server machines is potentially available to users on every DFS client machine in the DCE environment.

In DFS, we can access files stored on different machines in the computer network as easily as we can access files stored on our machine's local disk. We do not have to know the physical location of a file.

A given file can be stored on any DFS file server in the network. We just request a file by its DCE pathname and DFS finds the correct file location automatically.

2.2 DFS Machine Roles

There are different types of servers in a DFS environment. Each of these servers can handle one or more roles. These are called machine roles and were previously described in Chapter 1. We will now talk at length about the different machine roles in DFS. Note that we can assign different roles for each machine in a DCE cell environment. In order to do that, all that is necessary is to configure the appropriate processes that define what role the machine can play. For the RISC/6000, this can also be accomplished by using SMIT.

Machines roles are not mutually exclusive. Any server machine can assume multiple server machine roles. All server machines can be configured as DFS client machines, and any client machine can potentially be configured as a DFS server machine.

The following is a list of machine roles in a DFS environment:

- System Control Machine (SCM)

- Binary Distribution Machine (BDM)
- File Server Machine
- Fileset Database Machine
- Backup Database Machine
- Tape Coordinator Machine
- DFS Client Machine
- Private File Server Machine

2.2.1 System Control Machine (SCM)

The System Control Machine (SCM) controls various lists of users (and groups) that can perform administrative functions on the different types (machine roles) of DFS servers. The SCM houses the master copy of these lists that then are distributed to the various servers. The servers these lists get sent to are defined as the administrative domain. This domain is the set of machines that can be controlled by the same set of system administrators. There can be multiple domains in a DCE cell. A DFS server is only a member of one domain at a time. There is one SCM machine per administrative domain.

The SCM machine is the first machine that must be configured for any new domain. It can 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. On the RISC/6000 implementation of DFS, the /opt/dcelocal/var/dfs directory is linked to the /var/dce/dfs directory.

There are two processes that always run on an SCM. They are the *upserver* and *bosserver* processes.

The *upserver* process controls the distribution of administrative lists to all other server machines in the domain.

The upserver process communicates with a process called an *upclient* process. This process runs on a DFS server and contacts the upserver process that is running on the SCM for it′s domain. By default, the upclient processes on all DFS file server machines contact the upserver process on the SCM every five minutes to verify that it has the most recent version of each administration list. If the list is not the most recent, the upclient process on each machine retrieves the most recent version from the SCM and installs it locally.

The *bosserver* process is the Basic OverSeer Server process and it 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 processes that are listed in the BosConfig file. It restarts the failed processes automatically if they are listed in BosConfig file. The BosConfig file is located in the /var/dce/dfs directory.

The bosserver process also is used to create and maintain administrative lists and turn DFS authorization on and off. Administrative lists are also located in the /var/dce/dfs directory and can consist of the following depending on what machine roles are configured:

- admin.up
- admin.bos

- admin.fl
- admin.ft
- admin.bak

The admin.up list is used to identify all server principals that can obtain updates from the SCM. The list should include the principal names of all of the server machines in a domain. The bosserver process starts when DFS is started. We will discuss the other admin lists as appropriate throughout this book.

The bos command suite sends requests to the *bosserver* process. It is used by administrators to control access to principal and groups that are on the administrative lists and whether DFS authentication is activated on DFS file servers. Only principals and groups that are listed in the admin.bos administrative list can execute the bos commands.

Note

The bos commands are the most secure commands of the DFS. If you have authority to execute the bos commands, then you can do whatever you want in the DFS file system. As such, this list should only be used for the most trusted principals in your domain.

Figure 7 on page 16 shows the communication between an SCM machine and a DFS file server for a cell that has been defined to have two domains. You may have as many domains as you want in a cell but each domain can have its own set of administrators. This also implies that it must have its own SCM.

Generally, one administrative domain is sufficient.

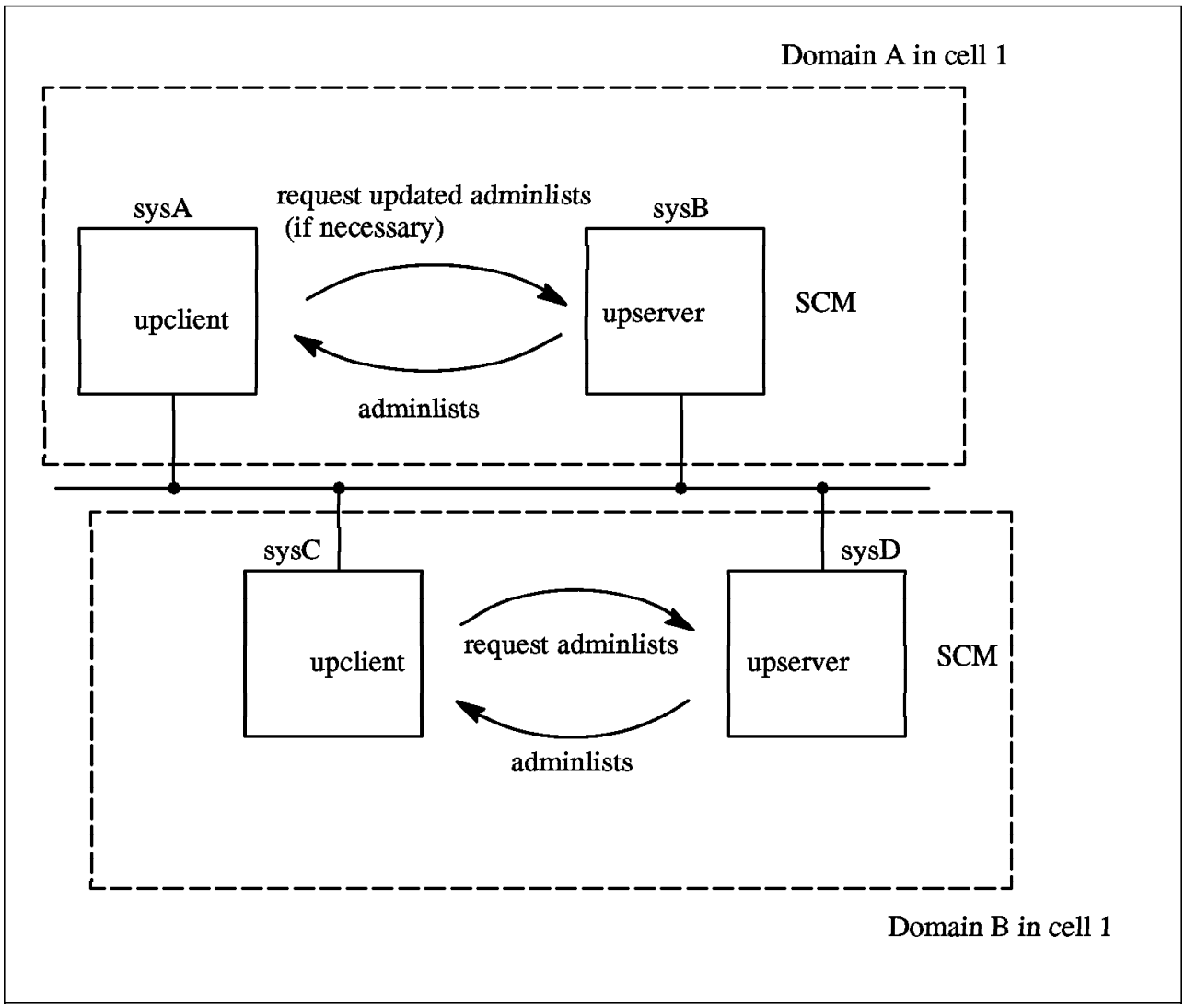


Figure 7. Each SCM Has its Own Set of Servers to Update With admin Lists

All of the above file server machines are assumed to be in the same cell. The upserver process for administrative list distribution is running on sysB and sysC so they are considered SCM machines. The upclient process periodically checks with the upserver process on the SCM in its domain to ensure that administrative lists for DFS servers are up-to-date.

2.2.2 Binary Distribution Machines (BDM)

Binary Distribution machines use the same upserver/upclient utilities as the System Control Machine. It stores DFS binary files for processes and command suites for distribution from its */opt/dcelocal/bin* and related directories to all other server machines of its CPU/OS type in a cell. Note that the /opt/dcelocal/bin directory is linked to the /usr/lpp/dce/bin directory on the RISC/6000.

The processes on Binary Distribution Machines are used to maintain proper versions of binary program files.

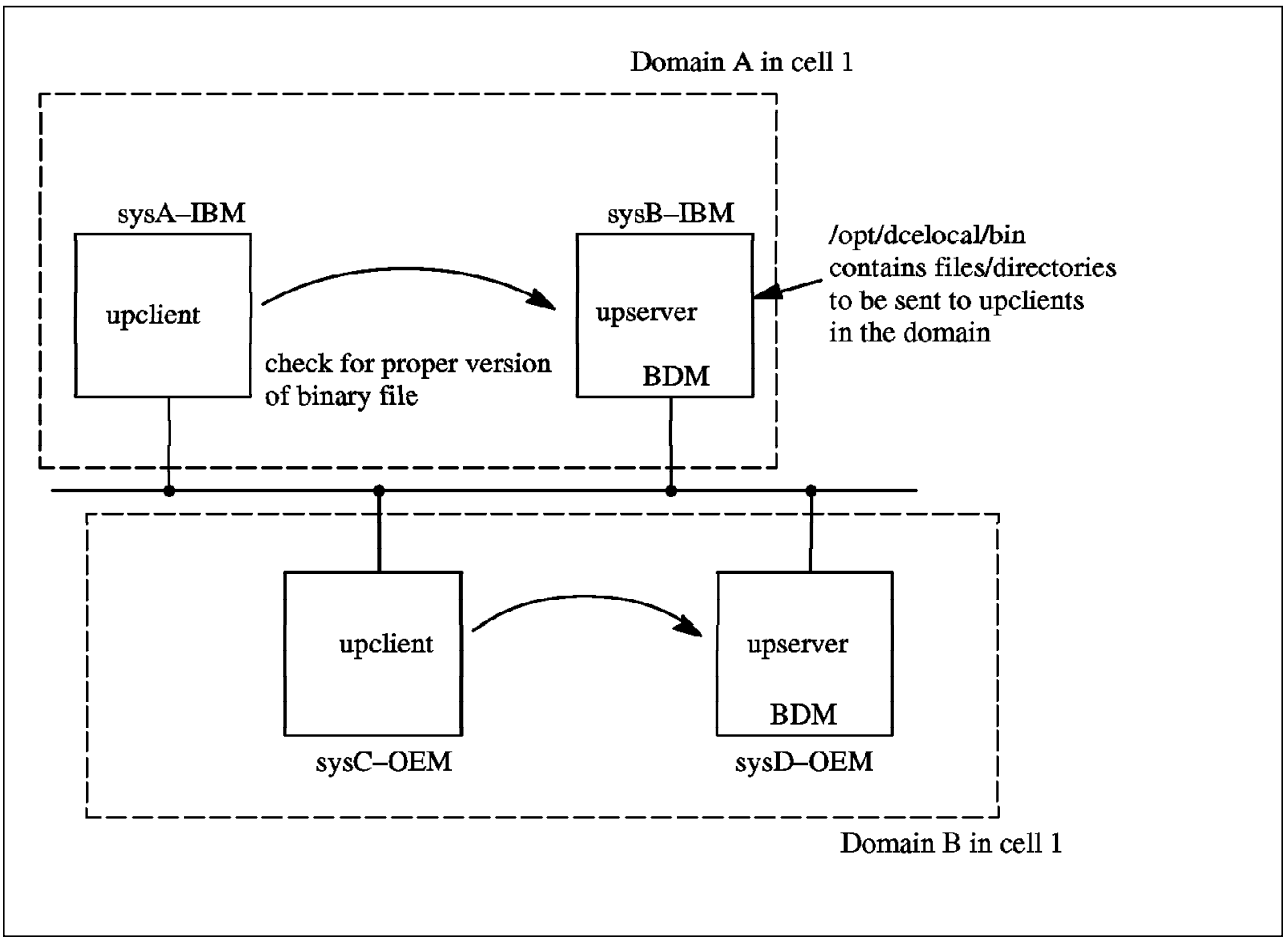


Figure 8. Each BDM Has its Own Set of Servers to Update With Binary Files

Figure 8 shows how Binary Distribution Machines can be used for upgrading the program binary files. These file are stored in the /usr/lpp/dce/bin directory.

Each DFS server keeps a copy of the DFS binaries in a local directory. This is usually the /usr/lpp/dce/bin directory for AIX on the RISC/6000. Ideally, you would want all the machines in your domain to be running the same version of the DFS binaries. The binaries are installed on a single BDM, which acts as a source for the others. We would have one BDM for each CPU/OS type. This would distribute the binaries to other machines in the domain that are of the same CPU/OS type.

A BDM runs the same upserver process as a SCM. The upserver process on a BDM helps ensure that all server machines of the same CPU/OS type in a cell are running the same DFS binary files. Unless a server machine is fulfilling the roles of both SCM and BDM, different upserver processes handle the distribution of configuration and binary files. A machine configured to perform both roles runs only a single Update Server (upserver) process to distribute both common configuration files and system binary files.

Note

The BDM can be used to distribute binaries that are not related to DFS.

Like the SCM, the BDM runs an upserver process. The upclient processes on the other server machines of the same CPU/OS type in the cell contact the upserver process by default every five minutes to verify that the most recent version of each binary file is in use. If it is not, the upclient process on the other server machines retrieve the most recent version from the BDM and installs it locally. Note that the source directory and filename *must* be the same as the target directory and pathname.

Note

For AIX DCE, the AIX install and update LPP history will not be updated when a machine retrieves new versions of the binaries from the BDM machine.

2.2.3 File Server Machine

A File Server machine is used to store and export DCE LFS or non-LFS data for use in the global namespace.

A non-LFS file system is the same as a JFS file system

In AIX, the non-LFS file system is the JFS file system. Throughout this book the use of non-LFS and JFS are equivalent.

In a cell, you may have many file servers configured. One of the benefits of distributed systems is that when you run out of filespace, all you have to do is add another file server machine. A File Server machine must run the following processes:

- **bosserver** - Basic OverSeerServer
- **ftserver** - Fileset Server
- **upclient** - Update Client for a set of admin lists or binaries
- **fxd** - File Server fileset exporter
- **dfsbind** - Interface to directory and security services
- **repserver** - Used to store ReadOnly replicas (optional)

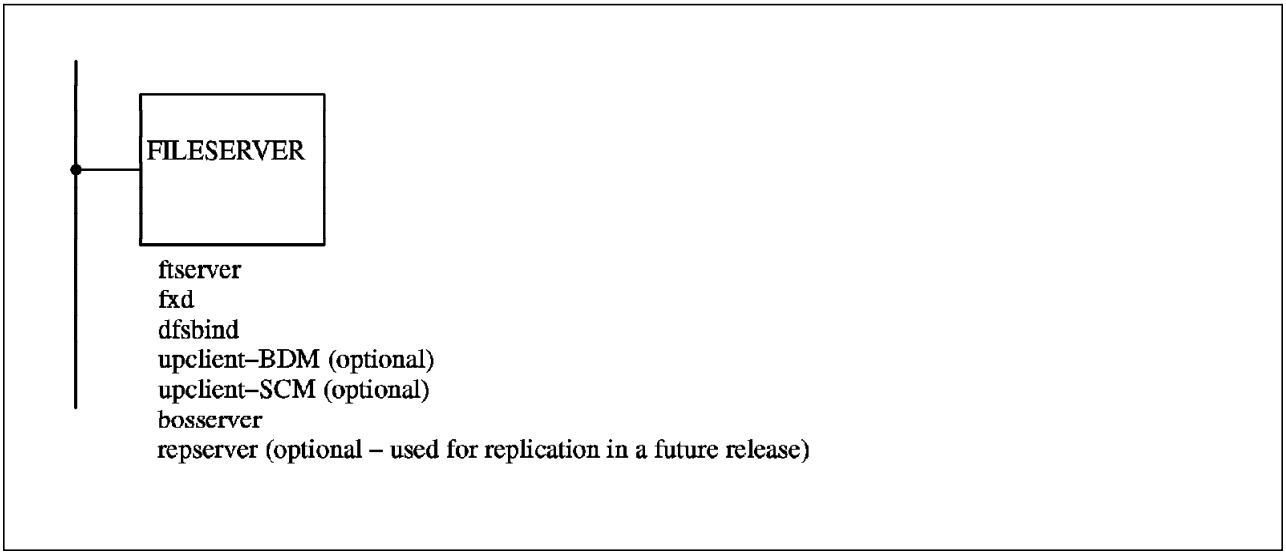


Figure 9. DFS File Server

If the file server is also a SCM or a BDM machine it will also run a *upserver* process. If the file server also has the machine role of a Fileset Location Database Server, it will run the *flserver* process.

The above processes are monitored by the bosserver process. The bosserver process restarts all failed processes automatically.

The ftserver and flserver processes provide a command suite called the *fts command suite*. This is used to operate on filesets. These `fts` commands are used for fileset creation, deletion, backup or to move a fileset to a different physical location.

The File Exporter is initialized by the fxd process in the kernel. The File Exporter runs as part of the kernel on each file server machine. It provides the same services across the network as the operating system provides for a local disk. These services include delivering requested files and programs, storing files, maintaining the directory structure, handling file and directory related requests, tracking status information and creating symbolic links between files and directories.

The command line for the *fxd* process 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 DCE Security Registry Database.

Members of this admin group can change permissions, ownership, and times of files and directories exported by the File Exporter. For files and directories in DCE LFS filesets the permissions include both ACLs and UNIX permission bits. For non-LFS filesets on UNIX permission bits apply. See section 3.4.2, “File Exporter” on page 59 for more details.

The replication of filesets will be available in a future release. When it is available, there will be a process called *repserver* that runs on the file server where replicated filesets are stored.

2.2.4 Fileset Database Machine (FLDB)

A Fileset Database machine stores the Fileset Location Database (FLDB). It runs the process (flserver) that maintains the FLDB. The purpose of the FLDB is to take a pathname for a file that is located in the DFS namespace and determine the location of the file server that has that file. The pathname request is from a DFS client cache manager. The end result is that the user never has to know the physical location of the file.

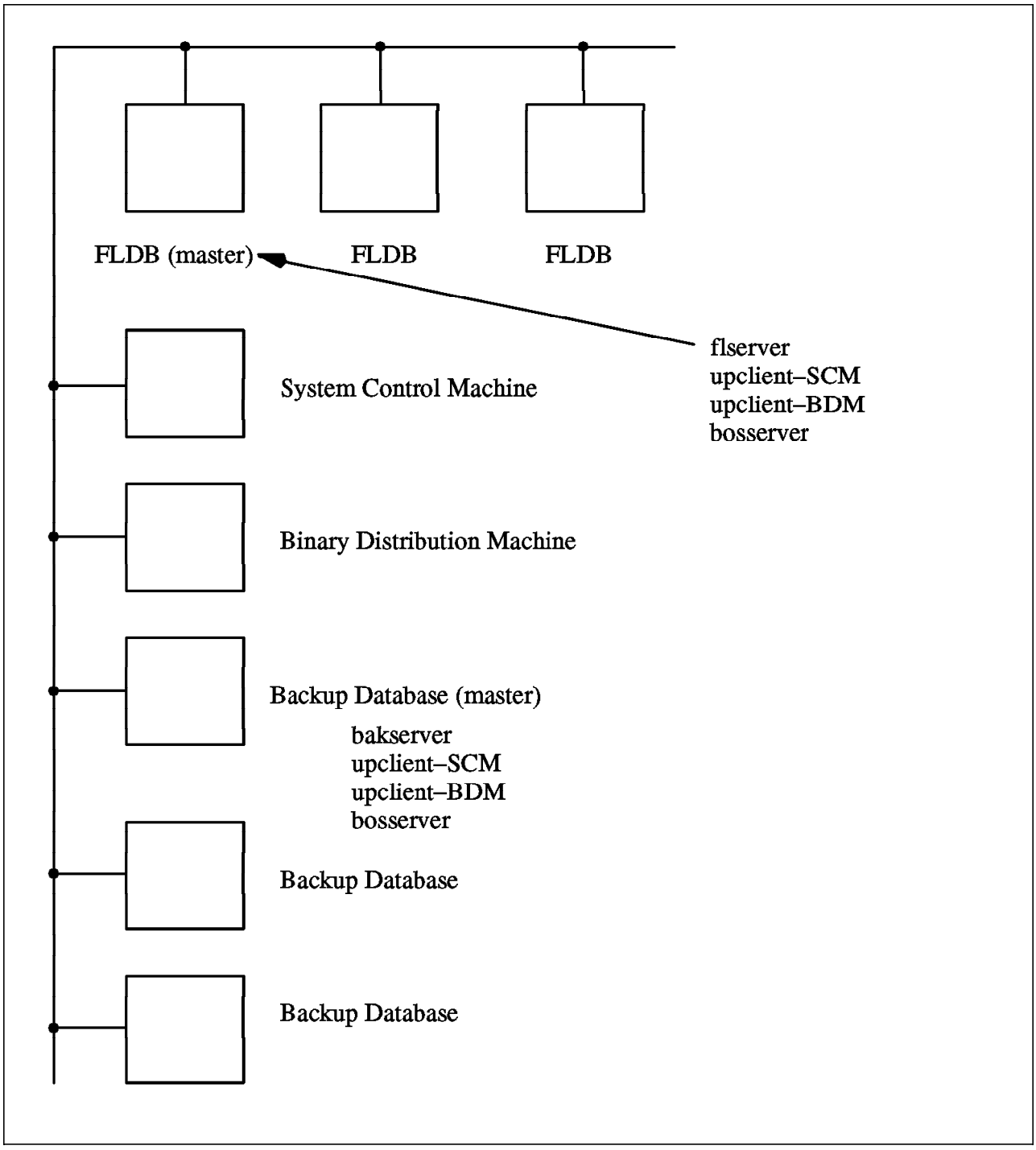


Figure 10. DFS File Location Database (FLDB)

Ideally, there should be three or a larger odd number of FLDBs configured in the cell. Having multiple FLDB servers eliminates a single point of failure for the FLDB. An odd number of Fileset Database machines is suggested because the FLDBs vote to see which FLDB is the primary site. The remaining FLDBs are secondary sites. The primary site is the FLDB to whom all changes are made while the secondary sites are read-only copies of the primary. Note that a DFS client cache manager may contact any FLDB for the location of a file. However, only the primary FLDB site may be updated with fileset location information. The primary site propagates the changes to the secondary sites. The FLDBs undergo a selection process to determine which FLDB should be considered to be the primary site. This happens through a library of utilities called *ubik*. These ubik utilities rely on the action of a quorum of vote. The ubik coordinator which runs on the primary FLDB site periodically sends an RPC to each secondary FLDB site and expects a response from the coordinator at the secondary FLDB site. This serves as a vote to maintain the current synchronization site for a fixed amount of time.

During this amount of time, a FLDB that was elected as the primary site sends subsequent RPCs to the secondary sites. It continues in the role of primary site confident that other sites have not chosen a new synchronization site. The primary site can then continue to make changes to the fileset location database. The primary site continues in this manner as long as it receives the votes of more than 50% of all FLDB databases (including itself).

If an even number of FLDB databases are available there is a higher weight to the vote which come from a database machine with the lowest network address of all database server machines. This allows ubik to attain a quorum.

Note that if a network partition occurs (your network breaks), the ubik technology on the FLDBs vote for another machine to take over as a primary FLDB site. The FLDBs update themselves to the latest FLDB information that is available and continue to operate according to the ubik algorithm.

Each Fileset Database machine runs the following processes:

- A Fileset Location Server *flserver* process. It is used to track the locations of all filesets in a cell and records changes in the FLDB. There is one master copy of the FLDB per cell.
- An *upclient* process: This is used to retrieve configuration files (admin lists) from the System Control machine.
- A BOS Server *bosserver* process. This is used to monitor other processes and to automatically restart failed processes as necessary.

A second upclient process is optional. This second process would be used to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.

2.2.5 Backup Database Machine

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. As with Fileset Database machines, you can have more than one of these machines in order to eliminate a single point of

failure. It is best to configure three or a larger odd number of Backup Database machines sufficient to backup the cell's data. These backup database machines use ubik technology in the same manner as the FLDBs. This means that there will be a primary site as well as secondary sites.

Each Backup Database machine runs the following processes:

- A Backup Server (*bakserver* process). This process maintains the Backup Database. The backup database is a collection of filesets that can be backed up at certain times. This process must be running on all machines that store a copy of the Backup Database.
- Two *upclient* processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server(*bosserver* process). This process monitors other processes and restarts failed processes automatically.

As with the FLDB, a second upclient process is optional. This second process would be used to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.

All details about Backup System will explained in section 3.5, " Backing Up DFS Filesets to Tape" on page 61.

2.2.6 Tape Coordinator Machine

The tape coordinator machine is used to physically back the filesets up to tape. It communicates with the backup database machine who tells it which filesets to backup and when this should be done. The tape coordinator machine can be the same machine as the backup coordinator machine.

2.2.7 DFS Client Machine

A DFS Client machine can be either a single or multiuser workstation. It communicates with File Server machines to access files for application programs, provides local data storage and does actual computation. Note that DFS servers can also function as DFS clients.

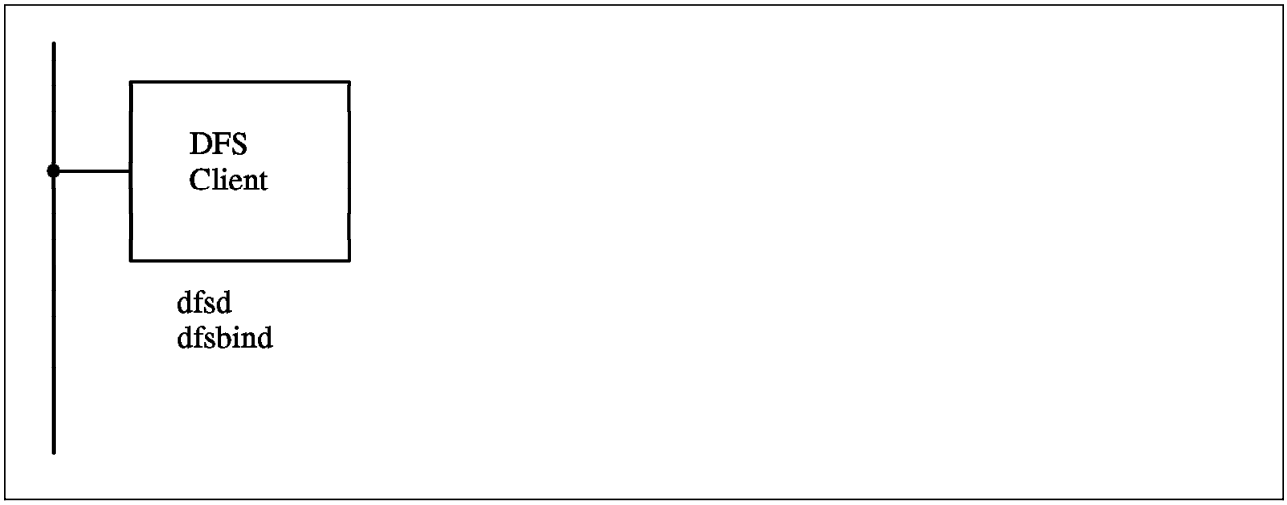


Figure 11. DFS Client

Each client machine must run the following processes:

- The *dfsd* process. This process initializes the cache manager in the kernel. The cache manager communicates with the server process running on File Server machines to obtain the location of files that the user has requested. It can be used to alter aspects of the cache manager's cache such as location and size. It also starts some background daemons that perform maintenance tasks. The cache manager improves performance by using write-behind caching of updated data. The cache manager also controls token daemons that respond to token revocation requests from File Exporters.
- The *dfsbind* process. This user-level process acts as an intermediary between the DFS kernel processes and DCE CDS/Security user-level processes. It runs on both DFS Client and File Server machines. The dfsbind process contacts CDS to resolve DCE pathnames that it receives from the DFS client; if it encounters a junction for the DFS file space, it returns information about the Fileset Location Database Machines for the cell. The dfsbind process is also used to obtain user authentication information.

2.2.8 Private File Server Machine

The primary function of a client machine is to communicate with File Server machines to access files. However, the client machine can also be configured as a Private File Server machine to export data from its local disk for use in the global namespace. The difference between a private file server and a regular DFS file server is that a private file server is administered by the local system administrator. Recall that the DFS file servers were administered on a domain basis. This was accomplished through the use of the admin.ft administration list.

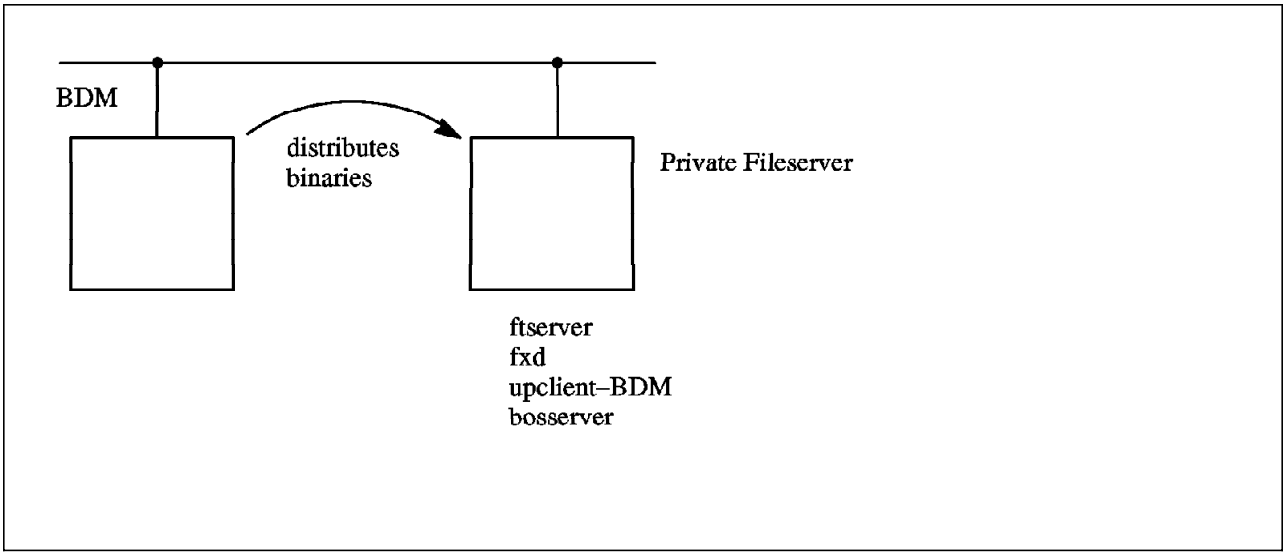


Figure 12. A DFS Private File Server

To export data as a Private File Server machine a client machine must meet the following additional requirements:

- Must be configured as a DCE client machine
- Must also be configured as a DFS client machine
- Have a server entry in the FLDB

- Run the Fileset Server (*ftserver* process)
- Run the File Exporter (initialized with the *fxd* process)
- Run the BOS Server (*bosserver* process)

You can optionally run the *upclient* process to retrieve binary files from the proper Binary Distribution machine

Although meeting these requirements qualifies a client machine as a File Server machine, that is not the machine's primary function. The machine is meant to function primarily as a DFS client machine where the administrator of that local machine can also export a subset of files and directories that other users can use. Once again, the difference is that it is only the local administrator that can administer this file server.

2.2.9 Summary of DFS Machine Roles

The following table is from the InfoExplorer* documentation on DFS. It summarizes the DFS machine roles and provides a brief description of its purpose. The DFS processes that each machine role must run are also listed.

The only prerequisite for these machine roles is that any machine must be first configured as a DCE client machine. Configuring these machines though SMIT for these machine roles will assure that they will have both an RPC binding in CDS for its pathname and a DFS server principal defined in the DCE Security Registry Database.

The methods used in implementing these various DFS machine roles are discussed in detail in Chapter 4, “Installing and Configuring DFS” on page 75.

Any server machine can be configured to perform any of the other server machine roles. A DFS file server machine is minimally configured as a DFS client machine. This enables a user on a file server machine to determine whether that machine has successfully exported its filesets into the DFS namespace.

Table 1. Summary of DFS Machine Roles

Machine Role	Purpose	Process	Suggestions
System Control machine	Distribute common configuration files	***bosserver*** ***upserver***(1) ***upclient***(2)	Use Binary Distribution 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***(1)	Use the System Control machine for a domain as a Binary Distribution machine
File Server machine	Export and store DCE LFS and non-LFS data	***bosserver*** ***ftserver*** ***fxd*** ***dfsbind*** ***repserver*** ***upclient***(1) ***upclient***(2)	A File Server machine must also have a server entry in the FLDB. In a large cell, dedicate one File Server machine to housing read-only replicas
Fileset Database machine	Store the Fileset Location Database(FLDB)	***bosserver*** ***flserver*** ***upclient***(1) ***upclient***(2)	Configure three Fileset Database machines. Configure Fileset Database machines as Backup Database machines
Backup Database machine	Store the Backup Database	***bosserver*** ***bakserver*** ***upclient***(1) ***upclient***(2)	Configure three Backup Database machines. Configure Backup Database machines as Fileset Database machines.
DFS Client machine	Server as a single-user or multiuser workstation; access files for application programs	***dfsd*** ***dfsbind***	

Note: The table uses the numbers 1 and 2 to differentiate the ***upserver*** and ***upclient*** processes running on the machines. The notation ***upserver***(1) and ***upclient***(1) denotes the Update Server that distributes common configuration files from a System Control machine. The notation ***upserver***(2) and ***upclient***(2) denotes the Update Server that distributes binary files from Binary Distribution machine.

2.3 Other Types of Distributed File Systems

There are several distributed file system implementations that are available today. Very widely used on IBM platforms and other vendor platforms is the Network File System (NFS) from Sun Microsystems**. Then there is the Andrew File System (AFS) originally developed at Carnegie-Mellon University and marketed today from Transarc Corporation. AFS is extensively used in universities and research organizations including over 20 IBM locations. We will compare and contrast these file systems with the Distributed File System (DFS). We will also show interoperability considerations in the event that you may want to use the AFS or NFS file systems with the DFS.

2.3.1 Comparison of the AFS and DFS

The Transarc Corporation submitted the distributed file system technology based on AFS to the Open Software Foundation (OSF). OSF selected AFS over other distributed technologies because it fulfilled criteria OSF defined for distributed file systems. This included being a general purpose file system and including next generation technology. Since AFS was enhanced and ported to use the DCE

core services and became known as DFS, there are some obvious similarities. We now look at some of the features of AFS and DFS. They are:

- Consistent naming for directories and files across all systems

  AFS and DFS allow users to address files with the same path name from anywhere in the network. This is an advantage when you have shell scripts or programs that you will execute from any machine in the network. The access to files and directories is transparent. The user does not need to know the physical location of a resource. The user only needs to know the pathname of the file or directory.

- Coherent management of the system from any location

  AFS and DFS requires at least one administrator per cell. The DFS administrator can work from any machine in the DCE cell and can perform most necessary actions on each of the participating DFS systems. The same is true for AFS. There are also sophisticated backup and restore methods built into the DFS technology. The DFS administration can also be accomplished on smaller groups of machines called domains.

- Protection and access control of resources

  AFS and DFS provide enhanced access control mechanisms called Access Control Lists (ACLs). Each file and directory is protected by access permissions. For DFS, these access permissions include read, write, execute, control, insert and delete for directories and read, write, execute and control for files. AFS ACLs uses different access permission types. DFS uses the DCE Security Service for user authentication and authorization information. Note that AFS only provides ACL protection at the directory level and not at the file level.

- High availability

  AFS and DFS allow for replication of ReadOnly filesets on separate systems. This makes directories and files available to users even if one of the ReadOnly replicas is unavailable.

  Note

  The current version of AIX/DCE does not support replication at this time.

- File and data integrity

  The AFS and DFS support a protocol that keeps the cache consistent. This results in that a client will work with the most recent version of a file even if there are multiple users using the same file.

- Scalability to large configurations

  Both AFS and DFS can scale to large configurations. There is no technical limit to the number of DFS clients that can access the DFS file system. The DFS can operate across very large networks. Data can be accessed across continents in the same manner as if it were located on a local machine.

- Good performance

  AFS and DFS support the concept of locally caching data once it has been transferred across the network. This eliminates the need to transfer data

multiple times and makes file access efficient when multiple accesses are required.

Although AFS and DFS are based on the same technology, there are differences. Some of the more notable differences are:

- DFS is fully integrated with other DCE Services. The DCE services that it makes use of are the Security Service, the Directory Service and the Distributed Time Service. AFS uses proprietary methods to accomplish this function. For example, in AFS the clock synchronization is performed by the Cache Manager. In DFS, the DCE Time Service provides clock synchronization.
- AFS is available today as a stable, mature product on many platforms. DCE is still a relatively new technology. DFS is shipping today on a relatively smaller subset of vendor′s platforms but growth is expected to be strong. Because DFS is shipped by the vendors themselves, usually it is more tuned to each platform than AFS is.
- DFS administration is similar to AFS administration. However, because of the underlying DCE layer an administrator must account for the complexities that are introduced by the DCE.
- DFS gives stronger client data cache consistency guarantees that AFS does. DFS supports POSIX single-site semantics which means that if a process running on one DFS client writes to a file, all other DFS clients will see the change on the next access of the file.
- DFS supports byte-range advisory locking while AFS does not.
- DFS allows you to export local AIX JFS file systems and still be able to access them via the local mount path. AFS does not.

2.3.1.1 Interoperability Between AFS and DFS

In general, AFS and DFS can coexist on a machine. In the following examples, we will assume the AFS Version 3.3 is being used.

Figure 13 on page 28 shows the case where a machine is configured as a DFS client and as an AFS client. Note that the machine will contain a cache for use with the AFS and a cache for use with the DFS. This configuration would exist in an installation where the client machine must be able to get to files that could be resident on either AFS or DFS servers. The client machine mounts the AFS file system on the /afs mount point. Therefore all files under the /afs directory come from the AFS namespace. In like manner, the client finds all of their DFS files under the /.../cellname/fs or /: directory as is the convention for a DFS client.

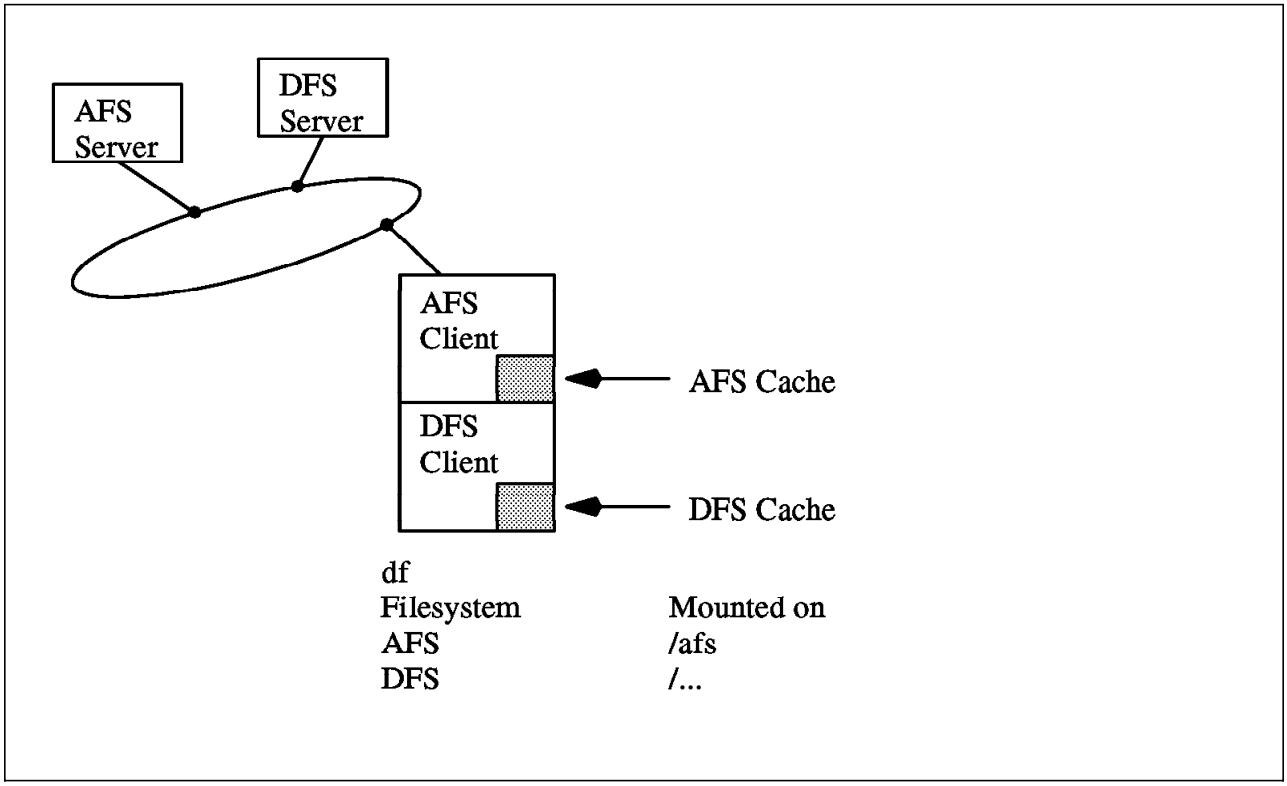


Figure 13. A Machine Can Be an AFS Client and a DFS Client

Figure 14 shows the case where a machine is configured as an AFS Server and as a DFS server. When configured in this manner, the AFS server must export different data than is being exported by the DFS server. In other words, the same files and directories can not be exported by both the AFS server and the DFS server. Note that in this figure, the AFS clients can only access data from the AFS server and the DFS client can only access data from the DFS server.

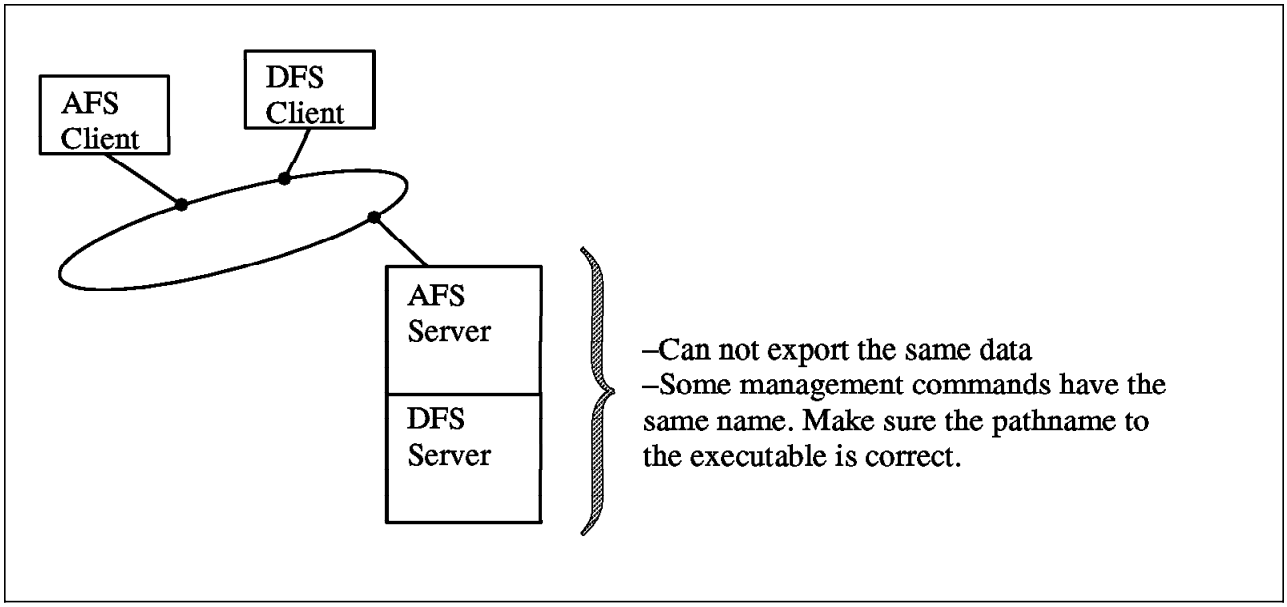


Figure 14. A Machine Can Be an AFS Server and a DFS Server

When an AFS installation is being migrated to a DFS environment, an AFS to DFS protocol translator can be used. This is shown in Figure 15 on page 29. The AFS to DFS translator function is available from Transarc. This translator function allows an AFS client to be able to access files and directories that are resident on a DFS server. Transarc also has some tools that can be used to assist in the migration from AFS to DFS. The function of these tools are also listed in Figure 15.

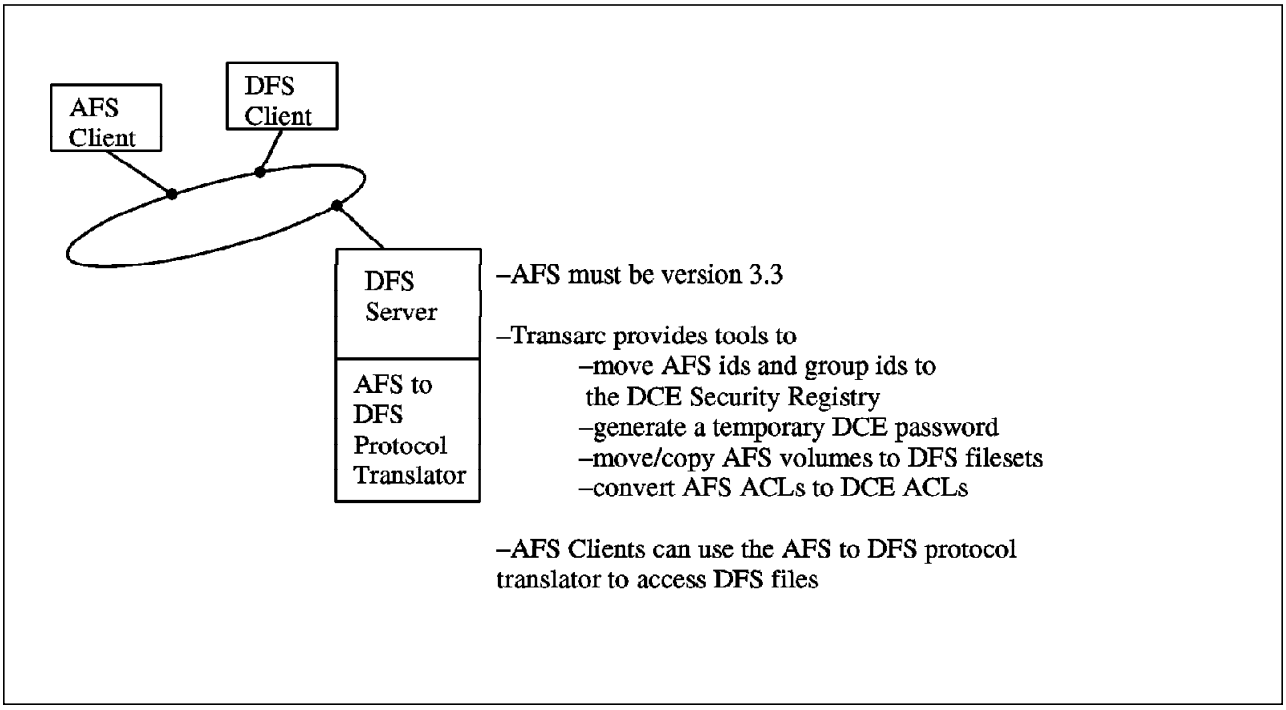


Figure 15. AFS to DFS Migration

2.3.2 Comparison of the NFS and DFS

Both the Network File System (NFS), originally from Sun Microsystems, and DFS are distributed file systems that allow users to access files and directories located on remote computers in a transparent fashion. From the user perspective, both allow users to work with remote files and directories as if they were local. NFS and DFS are compared and contrasted as follows:

- Remote Procedure Calls (RPCs)

  NFS uses Sun's ONC RPC. DFS uses the DCE RPC. Both NFS and DFS use UDP/IP as the underlying protocol layer.

- View of the Distributed File Space

  NFS clients construct their view of the distributed file system by explicitly mounting filesystems from the NFS server machines into their local file space. Each client can construct a unique pathname view to the distributed files.

  DFS maintains a global distributed file space which each DFS client mounts when the DFS client starts up. Each DFS client user sees the same pathname view of the files residing in the DFS file space. The DFS administrator

constructs the DFS file space by mounting DFS filesets into it. DFS clients do not have to know which DFS File Server machines are exporting what data, it is transparent to them. They do not have to explicitly mount directly from a DFS File Server.

- Coherent View of the Data from Clients

  DFS provides a coherent view of data from all DFS clients while NFS does not. This means that DFS supports the POSIX single site semantics. In other words, if data is changed by one DFS client all other DFS clients will see the change on the next read of the file.

- Caching at the Client

  DFS clients make use of extensive data caching of both file data and attributes; the data may be cached to the local disk or in-memory. NFS only provides limited in-memory caching. As a result, DFS clients produce less network traffic and less load on the file server. This contributes to larger client-server ratios while maintaining performance.

- Secure File Access

  By default, NFS uses an unauthenticated RPC protocol for file access. It is easy to impersonate a user from any NFS client machine in the network and gain access to that user′s files. DFS is integrated with the DCE Security Services and uses an authenticated RPC protocol by default. In addition, DFS supports the DCE Access Control Lists.

- Configuration and Administration

  The configuration and administration of NFS is a fairly easy job in small to moderate size environments. DFS administration is more complex due to increased functionality but scales better in large environments. DFS has a more central administration model than NFS and requires less administration on the client.

- File Locking

  Both NFS and DFS support byte-range advisory file locking.

- Data Replication

  DFS supports read-only replication of DFS filesets for increased reliability and scalability. NFS does not support data replication.

2.3.2.1 Interoperability between NFS and DFS

Many customers are currently using Network File Systems (NFS) as a technology to distribute file systems to remote sites. For these customers it would be helpful to allow access to the DFS filespace from systems which are not part of the DCE cell or which have no DFS code configured. AFS has a feature called an NFS/AFS Translator**, which allows NFS client systems to access the AFS filespace through the NFS/AFS Translator machine. This concept allows users on the NFS client system to access all files and directories (assuming the NFS client has the proper permissions) that are contained in the AFS filespace and exported by the NFS server system. See the following figure to understand the overall functions of the NFS/AFS Translator.

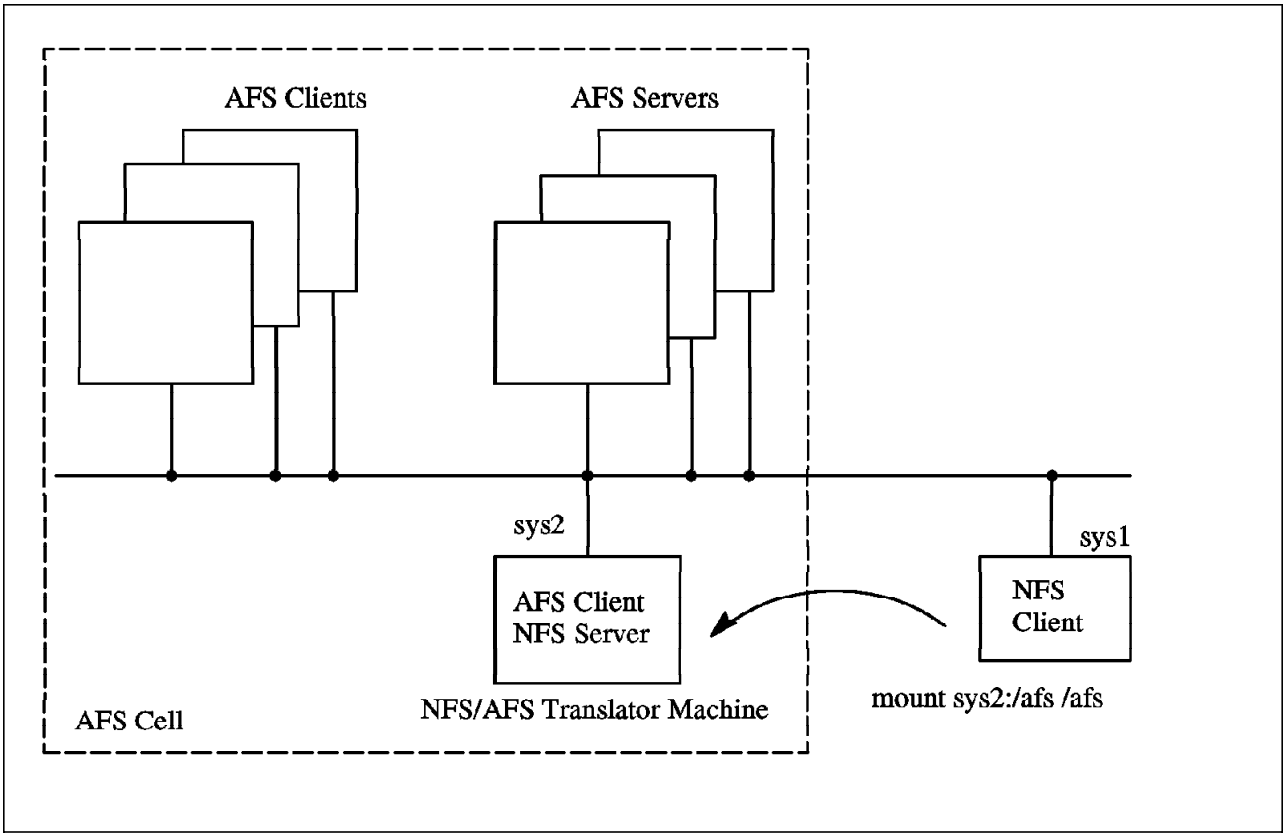


Figure 16. NFS/AFS Translator Function

Figure 16 shows that the NFS client has access to the entire AFS filespace through the use of the NFS/AFS Translator. The permissions that the NFS client has is that of *anonymous* if the user has not authenticated to AFS. The user can also authenticate to AFS and get specific user permissions. In order to do so, he must have an additional NFS/AFS executable installed on the NFS client.

There is not yet a NFS/DFS Translator function provided with the DCE-DFS products. However for the AIX DCE Base Services/6000 Version 1.2 release (and also AIX DCE Enhanced DFS/6000 Version 1.1 release), you may have NFS and DFS coexisting. There are two cases of interest. The first case is where a DFS server is also an NFS server. This is shown in Figure 17 on page 32. The DFS server can export the same JFS fileset to both the DFS namespace and as an NFS file system. Changes made from DFS clients to files and directories on the JFS file systems exported by the DFS/NFS server will be seen by the NFS clients. The reverse is also true. The NFS client only has access to all files and

directories that the NFS Server exports. This is just standard NFS. The NFS client does not have access to files and directories in the DFS namespace.

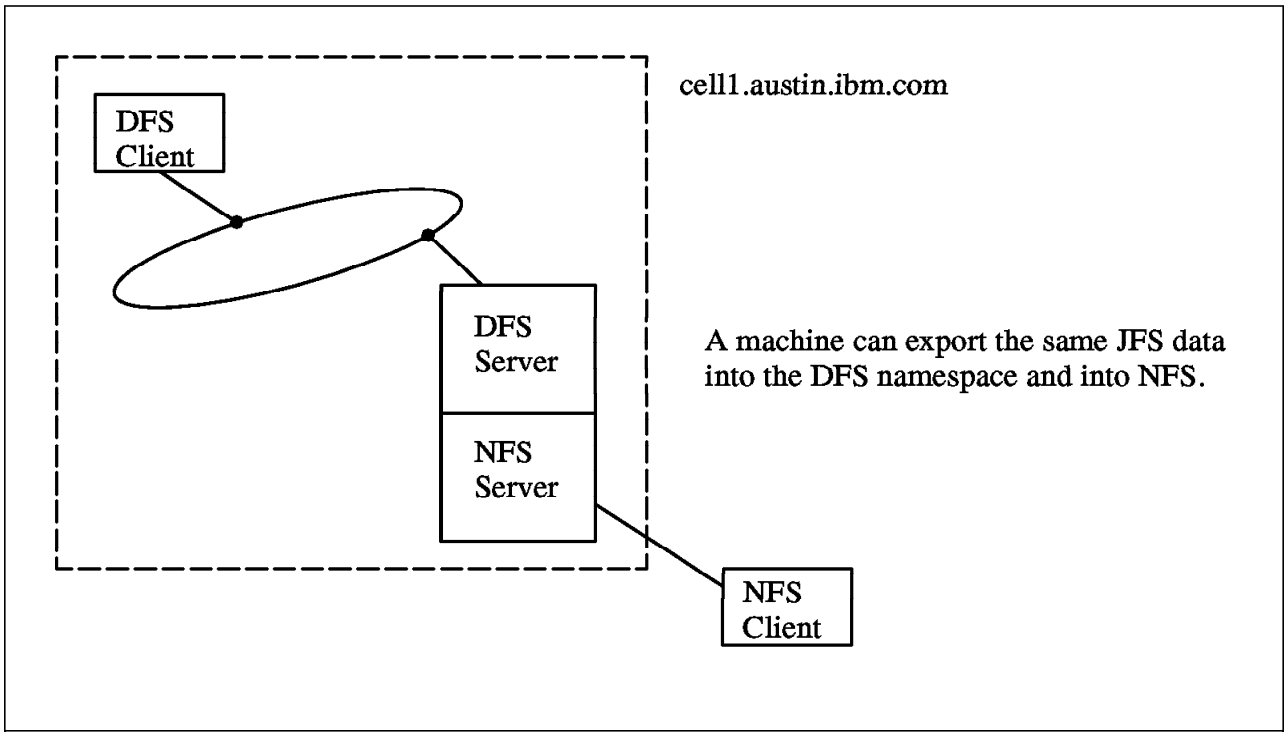


Figure 17. Same Data is Exported Into NFS and Into the DFS Namespace

The second case is where the NFS client is allowed to mount the entire DFS namespace. This is shown in Figure 18 on page 33. In this figure, sys2 which is a DFS client, exports the DCE root /... while the NFS client (sys3) mounts it. This allows an NFS client to have unauthenticated access into the DFS namespace. As such, this will give the NFS client the permissions that are associated with the *any_other* permission type.

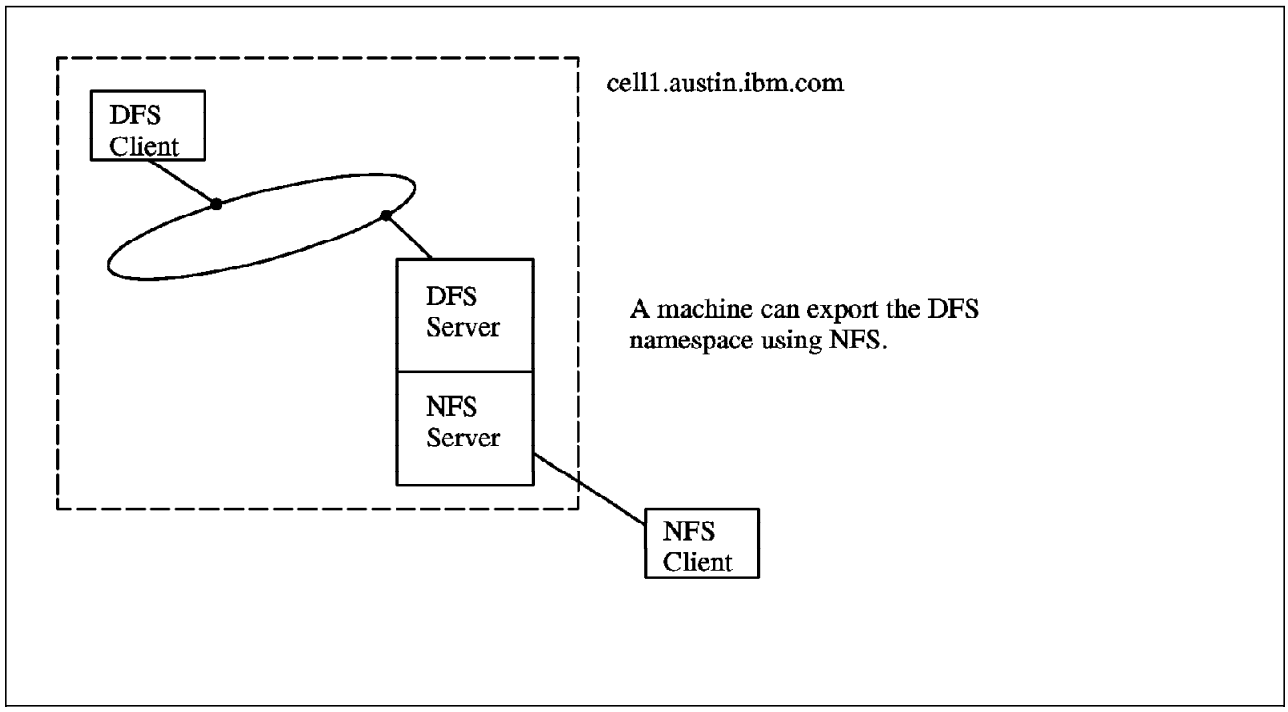


Figure 18. How an NFS Client Mounts the DFS Namespace for Unauthenticated Access

In a future release, an NFS to DFS authenticator will be provided. This will allow the user on the NFS client to log in to DCE on the NFS/DFS authenticator machine. This will result in the user being authenticated to DCE and he will have file permissions as an authenticated user. This is shown in Figure 19 on page 34.

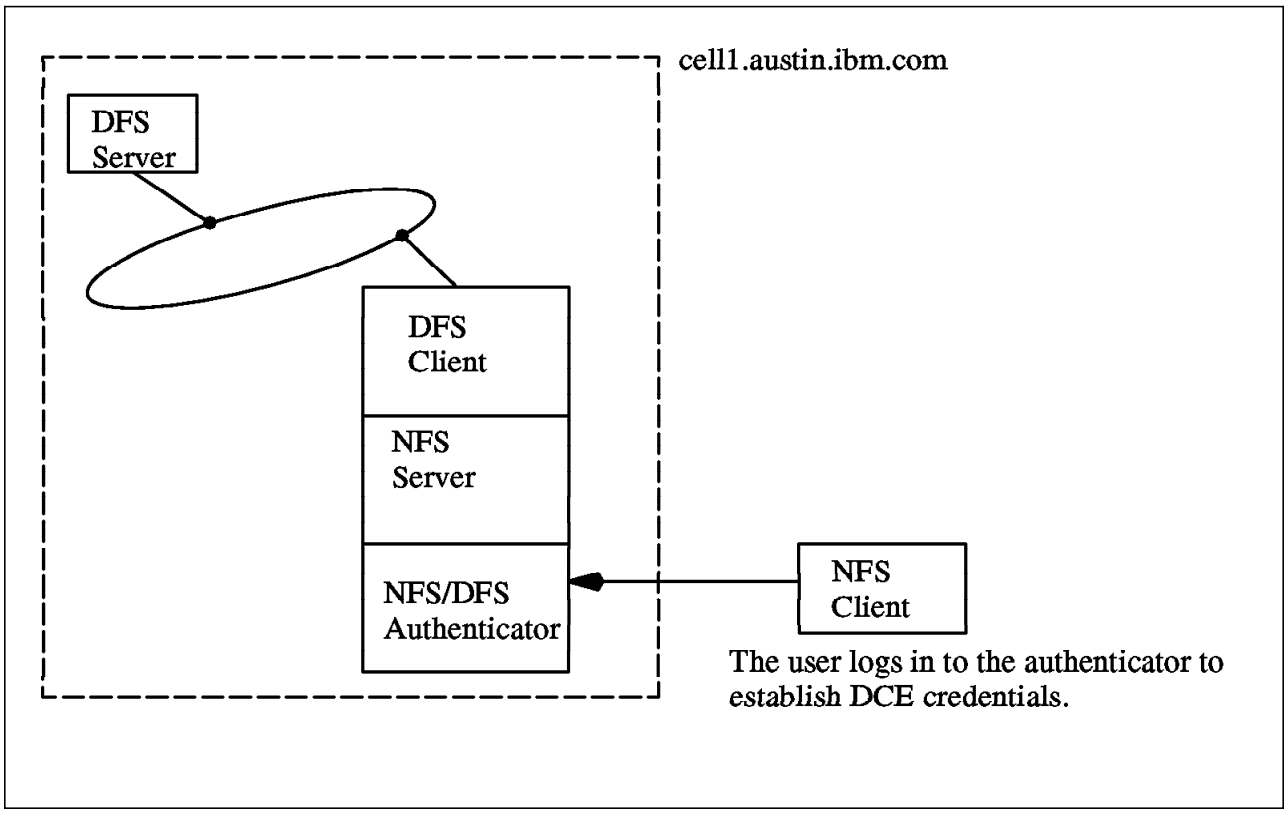


Figure 19. How an NFS Client Mounts the DFS Namespace for Authenticated Access

With the current release, there is a way to export DCE LFS filesets to NFS, but the procedure induces a security exposure. To export DFS files and directories into the NFS, you must use the acl_edit editor and provide the following ACLs on the files and directories that you want to export to NFS.

```
acl_edit /:/<directory> -m any_other:rwxcid
acl_edit /:/<directory> -ic  -m any_other:rwxcid
acl_edit /:/<directory> -io  -m any_other:rwxcid
acl_edit /:/<directory>/files -m any_other:rwxcid
```

Note that you do not have to grant all permissions. You only need to grant the permissions that the NFS client user's need. Frequently, this will be only the rx permissions.

The *any_other* ACL entry type must be changed on all files and directories that are to be exported to NFS. They must be changed so that they grant access to everybody. This removes any advantage of using DFS ACLs because now *all* the users have access to all the files and directories. This is not something that you would typically want to do.

Another scenario one might think of is to have access from the DFS filespace to file systems on an NFS Server system. A DFS client that is also an NFS client can mount into the NFS filespace but only the local system (the one which issued the mount of the NFS file system) can view the data. Other DFS clients cannot access the data. In Figure 20 on page 35, sys2 can access the files that sys5 has exported while sys1 can not. sys1 would have to NFS mount the file system exported by sys5 in order to use the files.

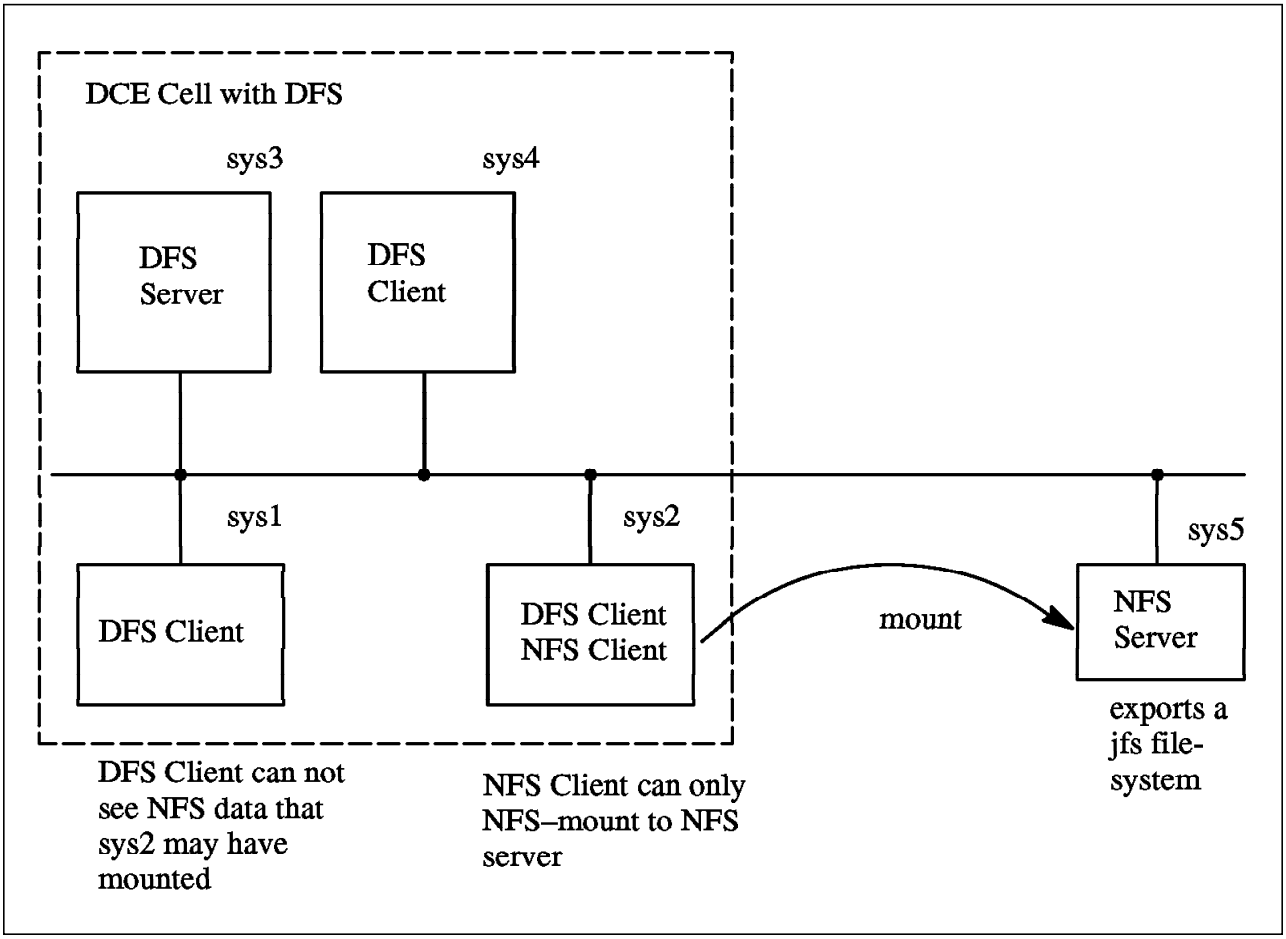


Figure 20. DFS Client Cannot See Changes to Files From an NFS Client

2.3.2.2 Migrating to DFS From AFS and NFS

It is possible to have AFS, NFS and DFS coexisting as you migrate your systems to DFS. Figure 21 on page 36 shows this interaction. In this figure, there is a DFS server and an AFS server. The AFS (version 3.3) clients retain their ability to access the AFS servers. They are also able to access the DFS servers by going through the AFS/DFS protocol translator. An NFS client (sys4 in this figure) is able to access the DFS namespace by mounting /... from a DFS client that is also acting as a NFS server. This results in the user on the NFS client having unauthenticated capability. For authenticated capability, the user on the NFS client would authenticate to the NFS/DFS authenticator that resides on the NFS server. (The NFS/DFS authenticator will be available in a future release.)

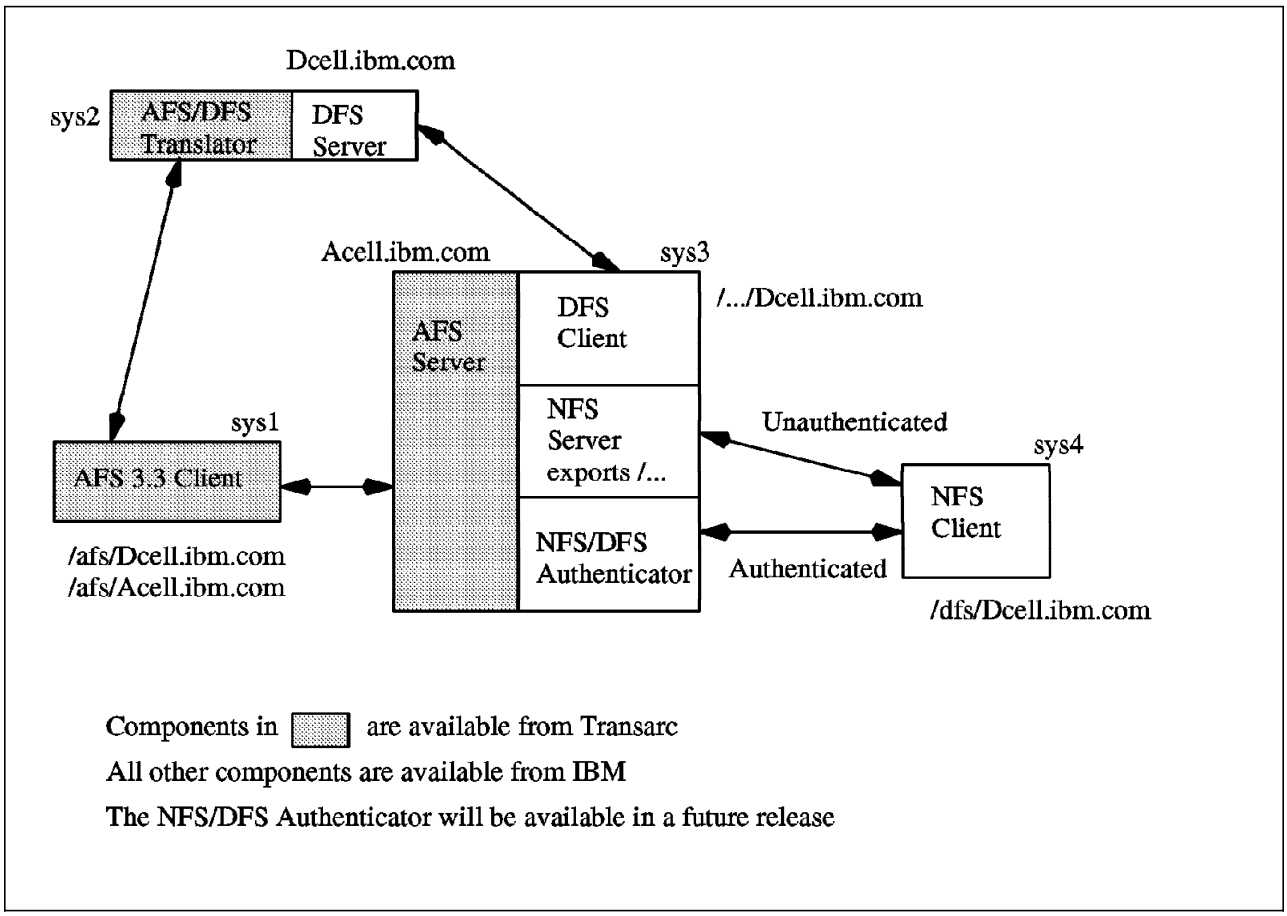


Figure 21. Migration to the DFS Environment

2.4 Planning Your DFS Configuration

This section is intended to assist you in planning for the establishment of *DFS Administrative domain*. A DFS Administrative domain is a collection of machines in the same DCE cell configured for administration as a single unit. Practically it means a group of machines in the same cell which retrieve the administrative lists from the same System Control machine. If you have a DCE cell which has a lot of users and data needed to be shared, you may require many DFS File Server machines. To simplify DFS management, you can group sets of DFS File Server machines into multiple DFS Administrative domains. Basically, the planning of DFS Administrative domains absolutely depends on its DCE environment. For the information on DCE, please refer to the ***DCE Administration Guide***. In this section, we will focus on the DFS Administrative domain considerations. This section describes:

- What to consider in determining DFS Administrative domains
- Practical guidelines for DFS Administrative domain

2.4.1 DFS Configuration Guidelines

The following are questions which you must consider when planning your DFS configuration.

- How do you expect your DCE cell to grow from the viewpoint of users?
- How much information does your environment currently have that needs to be distributed? How many users do you have in your network?
- What is the rate at which you modify data? Do the users in your network mainly look up information or do they create and change information at their systems?
- Do you need to use DFS replication? How much data do you need to replicate? On what machines should that data be stored?
- Where should you configure DFS client cache and what should the size of this cache be?
- Is communication limited to your own cell or do you need to communicate with other cells?
- If your cell spans geographically diverse regions or organizational boundaries, you might want to consider establishing multiple DFS administration domains. In general, you can start out with one administration domain and add other administration domains as needed.
- If there is a large volume of information that needs to be shared within your network, calculate the amount of disk space to be exported and the number of DFS File Server machines that you need.
- If information changes frequently and users in your network depend on the accuracy of that information, it is better to use a ReadWrite fileset. If users read or execute data but do not need to change (write) the data consider using data replication. You should only replicate filesets that contain read-mostly data.
- If you need to use DFS replication, you must create an DCE LFS root.dfs as a DFS root fileset. Only an DCE LFS root.dfs can support the DFS replication capability. You also need to consider that while replicating data helps availability, performance, and load balancing, there is a cost in terms of the amount of disk space required and the amount of administration required.
- If the machine will be configured as a DFS client and will have on-disk cache, you should create a new file system to hold the DFS cache files. The default directory of the DFS cache is */var/dce/dfs/cache* and the default cache size is 10 MB. We strongly recommend that you create a new file system to house the DFS cache.
- For your cell to communicate with other cells, you must define your cell in either a DNS or a GDS directory service and also have to have at least one GDA in your cell. It is also a good idea to pick your cell name to fit in either a DNS or GDS environment. For example, if you will be a DNS global nameserver - use a name like testcell.austin.ibm.com.

2.5 Planning the DFS File Space Structure

DFS provides the user in the DCE cell with a global view of a set of files and directories which were exported from multiple machines. It looks like a single namespace for the cell. In this section, we describe guidelines which assist you in setting up a well-structured DFS file space. The DFS file space hierarchy begins from */.../cell_name/fs*(or */.:/fs* , or */:*) junction. You can create and mount filesets in this file space hierarchy.

The first created fileset in a cell is the *root.dfs* fileset. Once *root.dfs* is created, it is automatically mounted on the */.../cell_name/fs* junction. After that you can expand the file space through creating and mounting other filesets. You can give filesets any names you would like to assign, but the recommended naming convention should:

- 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

Using a common prefix for related filesets is useful for the well-organized file space. The following figure is an example of how to structure a DFS file space:

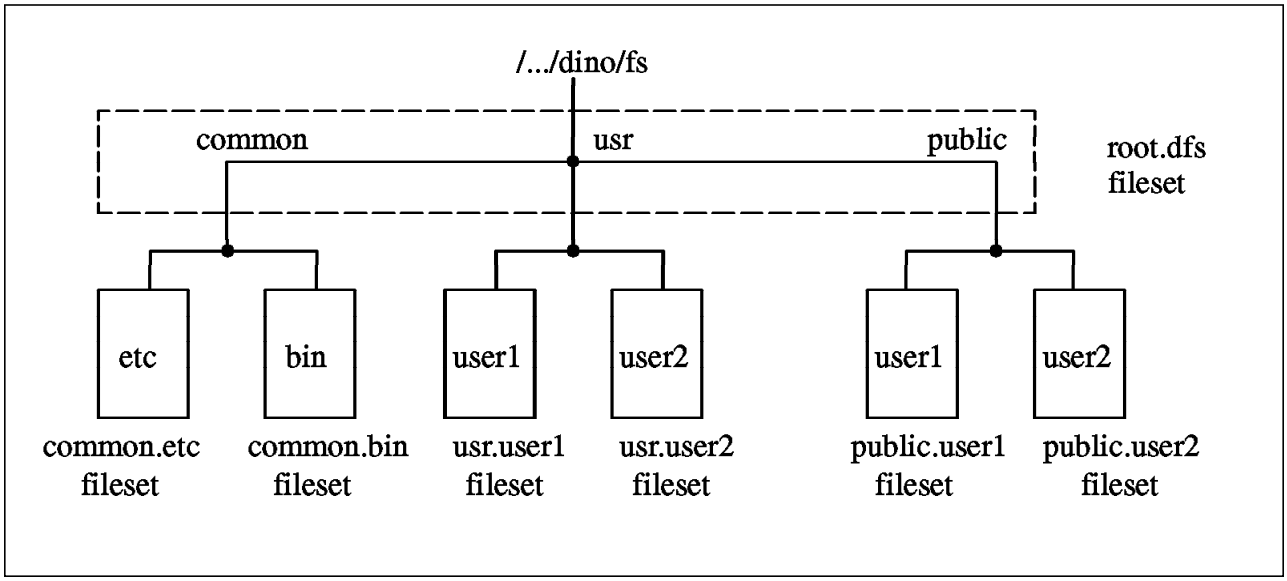


Figure 22. DFS File Tree

The above figure demonstrates the recommended conventions when you set up your DFS file tree. Each directory in */.../dino/fs* contains subdirectories that correspond to separate mounted filesets such as:

- The *common* directory

  This directory is the parent directory for the *common.type* filesets. For example, *common.etc* is for common configuration files and *common.bin* is for common executable files.

- The *usr* directory

  This is the parent directory for the *usr.username* filesets. This directory contains the home directory of each DFS user in the cell. For example, the

usr.user1 fileset contains */.../dino/fs/usr/user1* which is the home directory of *user1*.

- The *public* directory

  This directory is the parent directory for the *public.username* filesets. These filesets contain files that users want to make available to everyone. For example, the *public.user1* fileset for *user1* contains information that user1 wants to share with other users.

2.6 DFS File Location Database (FLDB) Scenarios

Depending on the number of DFS server machines in your cell, we can assume two kinds of DFS FLDB scenarios:

Single DFS FLDB

If your DCE cell has only one DFS server machine, this machine must run all processes and fill all required machine roles such as Fileset Location Database machine, File Server machine, and so on. In this case, you do not have to configure the System Control machine because you have just one version of configuration files (administrative lists). Make sure that the machine you configure as DFS server is big enough to run all processes required to be File Server machine. Be sure the machine you choose has enough space to store DFS filesets. The following figure is an example of single DFS server domain:

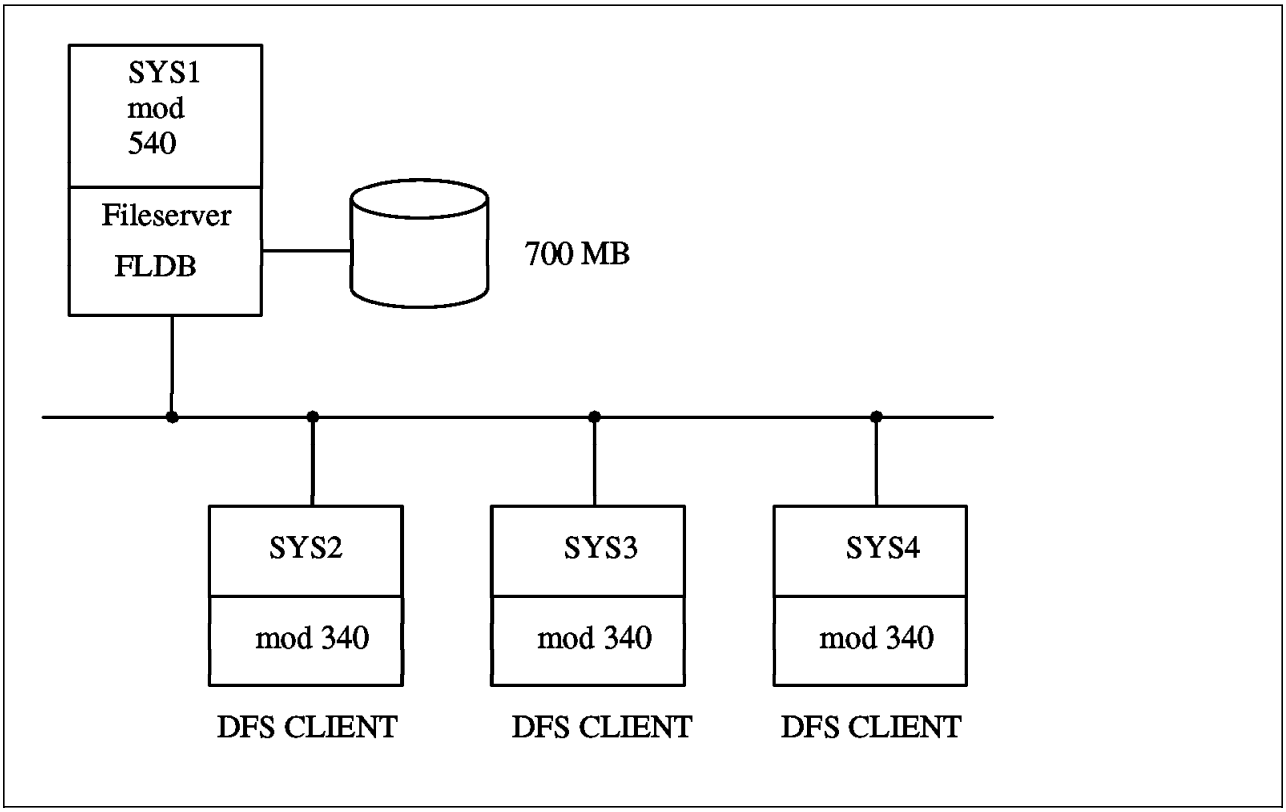


Figure 23. Single DFS FLDB

Multiple DFS FLDBs

In a DCE cell with three or more DFS server machines, it is recommended that you have three or more Fileset Location Database servers. This

configuration allows DFS to benefit from the database replication capability of DFS. An odd number of database machines is desirable if you have more than three FLDBs. There is a voting procedure to elect the master database server machine and an odd number of FLDBs guarantees that one will be selected as the master. Figure 24 is an example of multiple DFS servers domain:

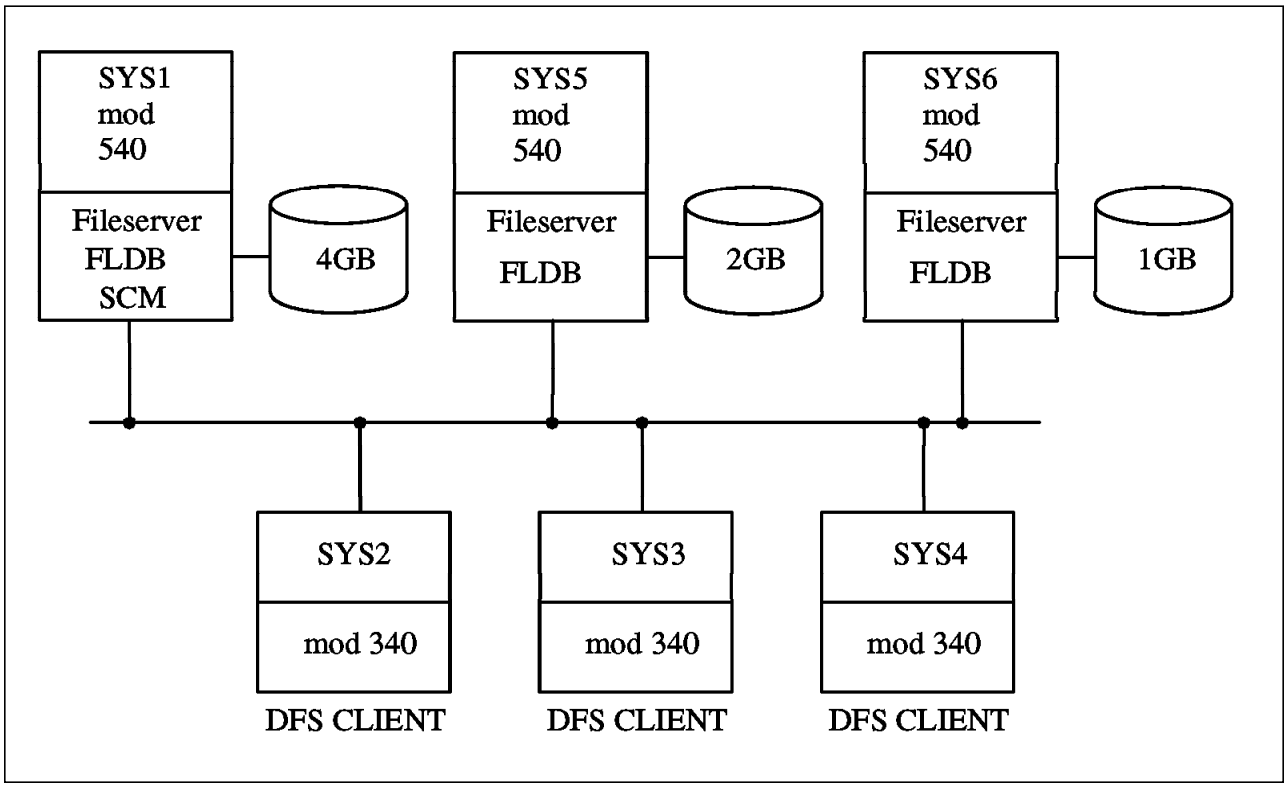


Figure 24. Multiple DFS FLDBs

Chapter 3. DFS Data Storage and Manipulation

DFS is a distributed application that manages information in the form of a file system. This chapter describes the units into which DFS data is organized and the active components that manage that data. This chapter also describes the benefits of the DCE Local File System as well as those provided by the non-LFS (JFS) file system.

3.1 DCE Local File System (LFS)

The DCE Local File System (LFS) is available with the AIX DCE Enhanced Distributed File System/6000 Version 1.1 product. The DCE LFS is a fast-restarting file system that integrates the capabilities needed for a large-scale distributed system with a sophisticated recovery mechanism. DCE LFS offers several capabilities not generally available in file systems such as support for logical groups of files (filesets) within a disk partition, support for ACLs and the ability to recover quickly from system failures.

DCE LFS is based on disk partitions and is integrated into the kernel. DCE LFS is designed to take advantage of a multi-threaded environment and asynchronous I/O. DCE LFS can be accessed both as a local file system when individual filesets are mounted and as a remote file system exported from file server machines. To provide uniform local and remote access, DCE LFS implements a compatible extension of the standard Virtual File System (VFS) functions.

Each DCE LFS fileset has a *fileset quota* associated with it. This allows a system administrator to assign a quantity of hard disk storage to the users of that fileset in a manner that best utilizes the physical hard disk space. The DCE LFS also supports the use of Access Control Lists (ACLs) to set permissions on directories and files in the DCE LFS filesets. ACLs extend the standard UNIX permissions (set with UNIX mode bits) to offer more precise definition of access permissions for users of directories and files.

3.2 JFS (or non-LFS) File System

A DFS file server can also export files from a non-LFS file system. For the RISC/6000, these files are based on the AIX JFS file system. The capability to do this is packaged with the AIX DCE Base Services/6000 DCE Version 1.2 product. By only using the non-LFS JFS file system, you will lose the capability of DCE ACLs and replication. We recommend that you use the DCE LFS file system if you think that you will need this capability in the future.

The remainder of this chapter will discuss both the DCE LFS and the non-LFS file systems.

3.3 Files and Directories, Filesets and Aggregates

DFS data is organized at three levels such as the one shown in Figure 25 on page 42.

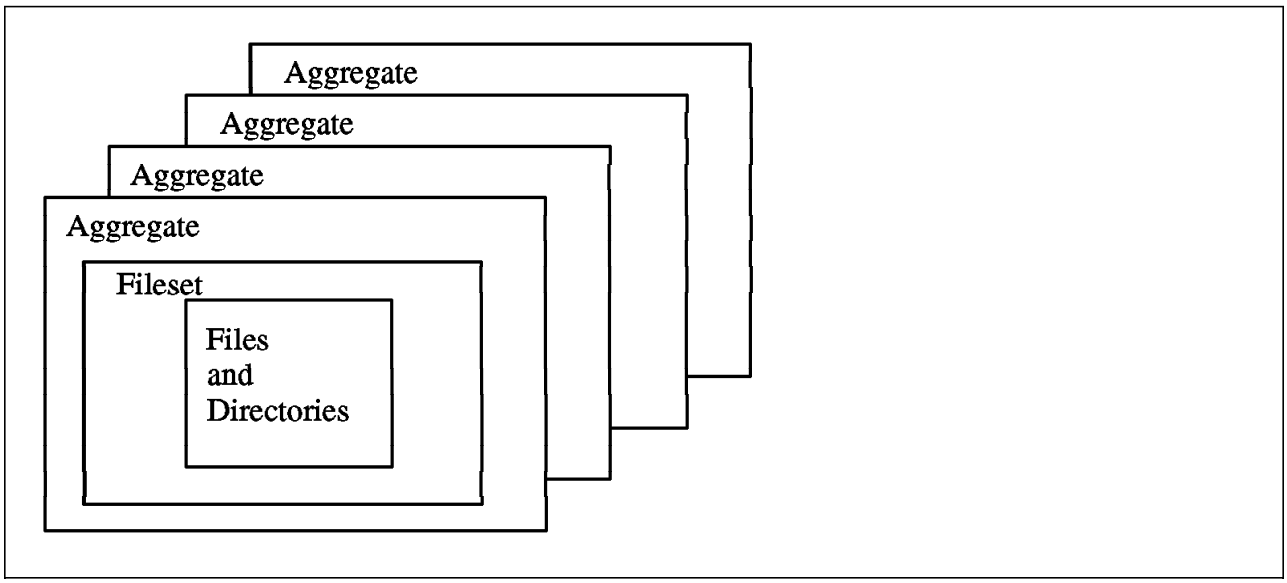


Figure 25. Files and Directories, Filesets, Aggregates

The three levels of DFS data organization are:

- Files and directories
- Filesets
- Aggregates

3.3.1.1 Files and Directories

The file is the unit that contains data. Directories organize files and other directories into a hierarchical tree structure.

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

In DFS, the permission associated with a DCE LFS file or directory are set using DCE Access Control Lists (ACLs). Permissions are set for non-LFS (JFS) files and directories with the standard operating system mode bits. DFS uses a specific implementation of DCE ACLs. These ACLs often referred to as DFS ACLs have read, write, execute, control, insert and delete permission bits. See Chapter 6, “Security” on page 161 for information on how to use ACLs.

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

- Specify correct pathname of where the file or directory is located in the namespace
- Be authenticated to DCE (for files that have specific permissions in their ACL)
- Be authorized to have access (via ACLs) to the desired files and directories

3.3.1.2 Filesets

A fileset is a subtree of files and directories that is smaller or equal to a disk partition. The fileset is a convenient grouping of files for administrative purposes. For example, the subtree of files pertaining to a particular project can be grouped on the same fileset. Filesets can either be DCE LFS or non-LFS (JFS) types. DCE LFS filesets have advantages over non-LFS filesets:

- They can be replicated.
- They can have Access Control Lists (ACLs) for a more granular protection.
- They can be moved from one machine to another while a user is actually using it.
- There can be multiple filesets on a disk partition.

By comparison, a non-LFS fileset is equivalent to a UNIX partition (in AIX, this is a logical volume). A non-LFS partition can house only a single file system (non-LFS fileset).

Both the DCE LFS and non-LFS filesets are attached to the DFS namespace by using a process similar to mounting a UNIX or AIX file system.

1. **Fileset Identification**

   Fileset names have to be unique within a cell and are assigned by the system administrator. This is true of both DCE LFS filesets and non-LFS filesets.

   Fileset names are stored in the FLDB (Fileset Location Database) and they may have a maximum of 102 characters (excluding .readonly or .backup).

   Valid characters are:

   - All uppercase and lowercase letters (a through z, and A through Z)
   - All numerals (0 through 9)
   - . (period)
   - - (hyphen)
   - _ (underscore)

     It also has to include at least one alphabetic character or an underscore (to differentiate it from fileset IDs).

     DCE LFS filesets can vary in size but they are always less than or equal to the size of an aggregate. Multiple DCE LFS filesets can be stored on a single aggregate.

     This allows system administrators to maximize machine and disk usage.

2. **Fileset ID numbers**

   Every fileset has a *fileset ID number*. The fileset ID number *and* the fileset name are both unique within the cell in which the fileset resides. When a fileset is registered in the FLDB the FLDB 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, we can omit the integer before commas if it is a 0 (zero); commands that accept a fileset ID number assume 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*.

Many DFS commands which operate on filesets allow you to specify either the fileset name or the fileset ID since both of these are unique within a cell.

3. **Fileset quota**

   Both DCE LFS and non-LFS filesets have a fileset quota assigned by the system administrator. A fileset′s quota determines the maximum amount of the data that can be stored on the fileset. For DCE LFS filesets, the quota defines the amount of disk space the fileset can grow to on the aggregate on which it is stored. For non-LFS filesets, the quota is always equal to the size of the partition on which the fileset resides.

   Fileset quota is measured in 1-kilobyte (1024-byte) units. The default quota for every DCE LFS fileset is 5Mb. It can be changed by the system administrator.

   Different filesets can have different quotas. If we wish to copy files from a directory on our fileset to a directory on another user′s fileset, we should remember that other user′s fileset may have a smaller quota than ours and may not have enough space to accommodate the file.

   Figure 26 on page 45 shows how a fileset′s quota limits the amount of data that can be copied to it.

   The figure highlights the DCE LFS filesets of two users, *pete* and *takeda*. Both filesets have a quota of 5000 1-kilobyte blocks. This is the default associated with every new DCE LFS fileset.

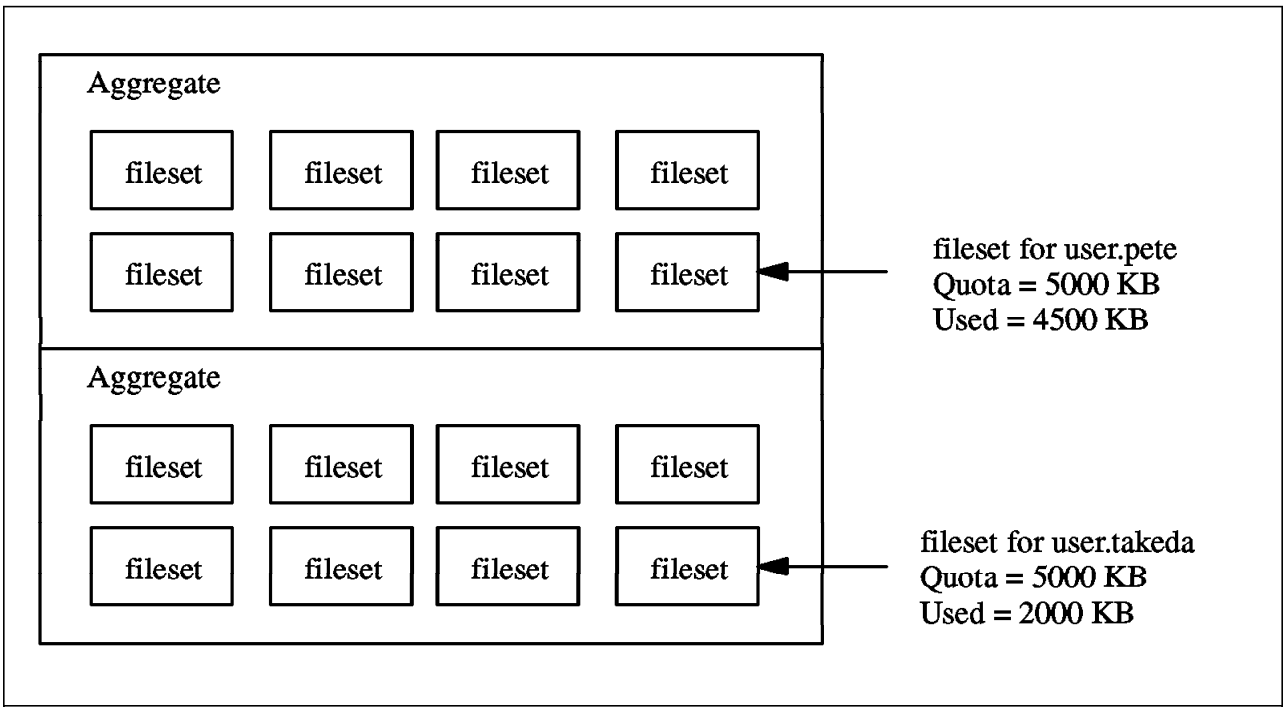


Figure 26. Fileset Quota

A file up to 3000 kb in size can be copied to user.takeda (5000 quota - 2000 used = 3000 available).

We can use the `fts lsquota` command to determine the quota of our fileset. We can indicate the name (or fileset ID number) of the fileset or by specifying the name of a file or directory located on the fileset. We can also 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 LFS and non-LFS filesets. The *fts lsquota* command displays the following information about DCE LFS filesets:

- The name of the fileset. For users, this is often the username, for example, user.pete or user.takeda
- 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 quota currently in use on the fileset.
- The percentage of available disk space currently in use on the aggregate where the fileset resides.
- The number of kilobytes currently in use on the aggregate and the total number of kilobytes.
- A DCE LFS indicator to show that the fileset is stored on a DCE LFS aggregate.

Here is the result of the output when an DCE LFS fileset quota is displayed:

```
# fts lsquota -fileset user.pete
Fileset Name          Quota    Used  % Used   Aggregate
user.pete              5000       9     0%     1% = 218/16376 (LFS)
```

The command displays the same information about a non-LFS fileset except there is a non-LFS indicator to show that the fileset is stored on a non-LFS partition. In this case, the fileset quota cannot be set. For AIX JFS file systems used with DFS, the fileset quota is equal to the size of the JFS file system and is only changed if the JFS file system is increased in size.

Here is the result of the output when a non-LFS fileset quota is displayed:

```
# fts lsquota -fileset user.takeda
Fileset Name          Quota    Used  % Used   Aggregate
user.takeda            8192     288    3%      3% = 288/8192 (non-LFS)
```

We need to check the quota of our fileset to verify that we have adequate space. If we are close to exceeding our fileset quota we may not be able to save changes to a file.

We are also not able to save a file if the aggregate that stores our fileset is full. This can occur even if we are not close to exceeding the quota on our fileset. An aggregate can be full even if one or more of the filesets it contains has quota remaining. This concept is called *overextending* . The system administrator is allowed to assign more space to filesets on a disk partition than actually exists. The system administrator detects when the aggregate is running out of room and can move a user's fileset to another aggregate. This results in being able to more efficiently use the room on a hard disk and it prevents allocating a user more room that he really needs. In Figure 27 user1, user2 and user3 have each been allocated a quota of 200MB. This results in a total usage of 600MB. The physical disk is only 450MB. When the system administrator detects that the aggregate usage is at a certain level, he can move the fileset belonging to one of the users to an aggregate that has more room (or he can extend the aggregate).

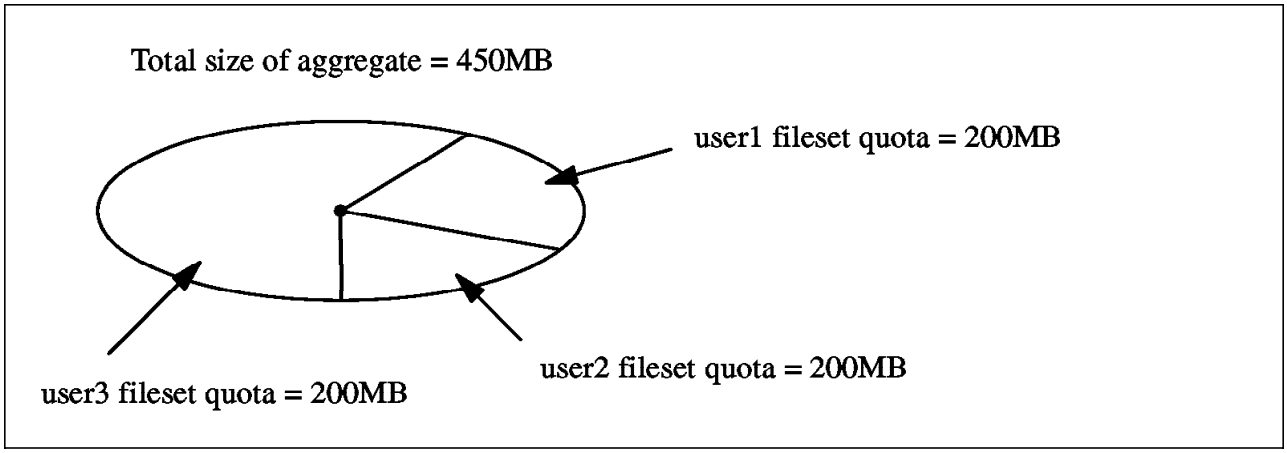


Figure 27. Overextending the Fileset Quota

4. **Three Different Types of DCE LFS Filesets**

   There are three types of filesets in the DCE LFS: read-write, read-only and backup. Non-LFS file systems do not have these different types of filesets. When used with DFS, non-LFS filesets are essentially treated as read-write filesets. However, a partition housing a non-LFS fileset can be marked as read-only in the local operating system. DFS treats it as read-only fileset and it cannot be modified. This fileset does not receive a *.readonly* extension.

Every DCE LFS fileset has a single *read-write fileset* 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 by way of 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 read-only replica is created. Each read-only fileset is given the same name as its read-write source with an additional *.readonly* extension. A read-only fileset uses up the same amount of storage as the read-write version.

A read-only fileset cannot be modified by commands such as `mkdir` or `rm`. 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.

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. This backup fileset takes up much less space than the original read-write version because it uses pointers to the same data blocks.

5. **A Fileset's FLDB information**

   The `fts lsfldb` and **fts lsft** commands display information from a fileset's FLDB entry. Each FLDB entry for a DCE LFS fileset contains the following information:

   - The name of the fileset, with a *.readonly* or *.backup* extension, if appropriate.
   - The fileset IDs of the read-write, read-only, and backup versions.
   - A separate status flag for each of the three versions, indicating whether the version exists at some site.
   - The number of sites at which a version of the fileset exists.
   - An indicator if the FLDB entry is locked. It 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 each read-only site, the MaxSiteAge replication parameter defined for that site.
   - 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).

   The following is an example of the `fts lsfldb` command:

   ```
   # fts lsfldb
   ```

```
(Some output deleted for clarity...)

user.pete
        readWrite   ID 0,,34  valid
        readOnly    ID 0,,35  invalid
        backup      ID 0,,36  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner
sys4                RW       lfs.users 0:00:00 hosts/sys4     <nil>

user.takeda
        readWrite   ID 0,,37  valid
        readOnly    ID 0,,38  invalid
        backup      ID 0,,39  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner
sys4                RW       /export/user2 0:00:00 hosts/sys4     <nil>
```

6. **Fileset's Header Information**

   A separate fileset header is stored at each site where a version of a DCE LFS fileset exists. The header is part of the data structure that records disk addresses on the aggregate where the files in the fileset are stored. This data structure is a method of grouping all of the files into logical units without requiring that they be stored in contiguous memory blocks. In addition, the header records some of the same information that appears in the FLDB. Therefore, even if the FLDB is unavailable, the `fts` commands can still access the information.

   The `fts lsheader` and `fts lsft` commands display information from a fileset's header. Each fileset header for a DCE LFS fileset contains the following information:

   - The name of the fileset.
   - The fileset ID number.
   - The type of fileset.
   - The storage space used by the fileset.
   - Additional internal information about the status.
   - The status flag for the site.
   - The File Server machine, aggregate name, and aggregate ID number.
   - 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.
   - A quota, indicating the maximum amount of disk space the fileset is permitted to occupy.
   - The day, date, and time the fileset was created.
   - The day, date, and time the contents of the fileset were last updated.

7. **Fileset Accessibility**

In any file system, two types of problems can affect our ability to access data: service outages and loss of data. DFS uses fileset replication and backup filesets to solve these problems.

In DCE LFS, each fileset has a read/write version. Data such as files and programs in a read/write version of a fileset can be modified. Fileset replication involves placing read-only copies of read/write DCE LFS filesets on multiple File Server machines. Data in a read-only version of a fileset cannot be modified. It can only be read and executed.

The replication of commonly used files greatly reduces the chance that these files will be unavailable.

3.3.1.3 Aggregates

We have already mentioned that a DCE LFS fileset is a hierarchical grouping of files managed as a single unit. A DCE LFS aggregate is a disk partition (logical volume) modified to include the DCE LFS metadata structure that supports DCE access control lists (ACLs), multiple DCE LFS filesets, logging and other fileset related operations. You can store multiple DCE LFS filesets on an DCE LFS partition.

Using DFS, we 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.

Figure 28 on page 50 illustrates the structural differences between the DCE LFS and other non-LFS file systems. The DCE LFS aggregates can store multiple DCE LFS filesets. The non-LFS partitions can store only a single non-LFS file system each.

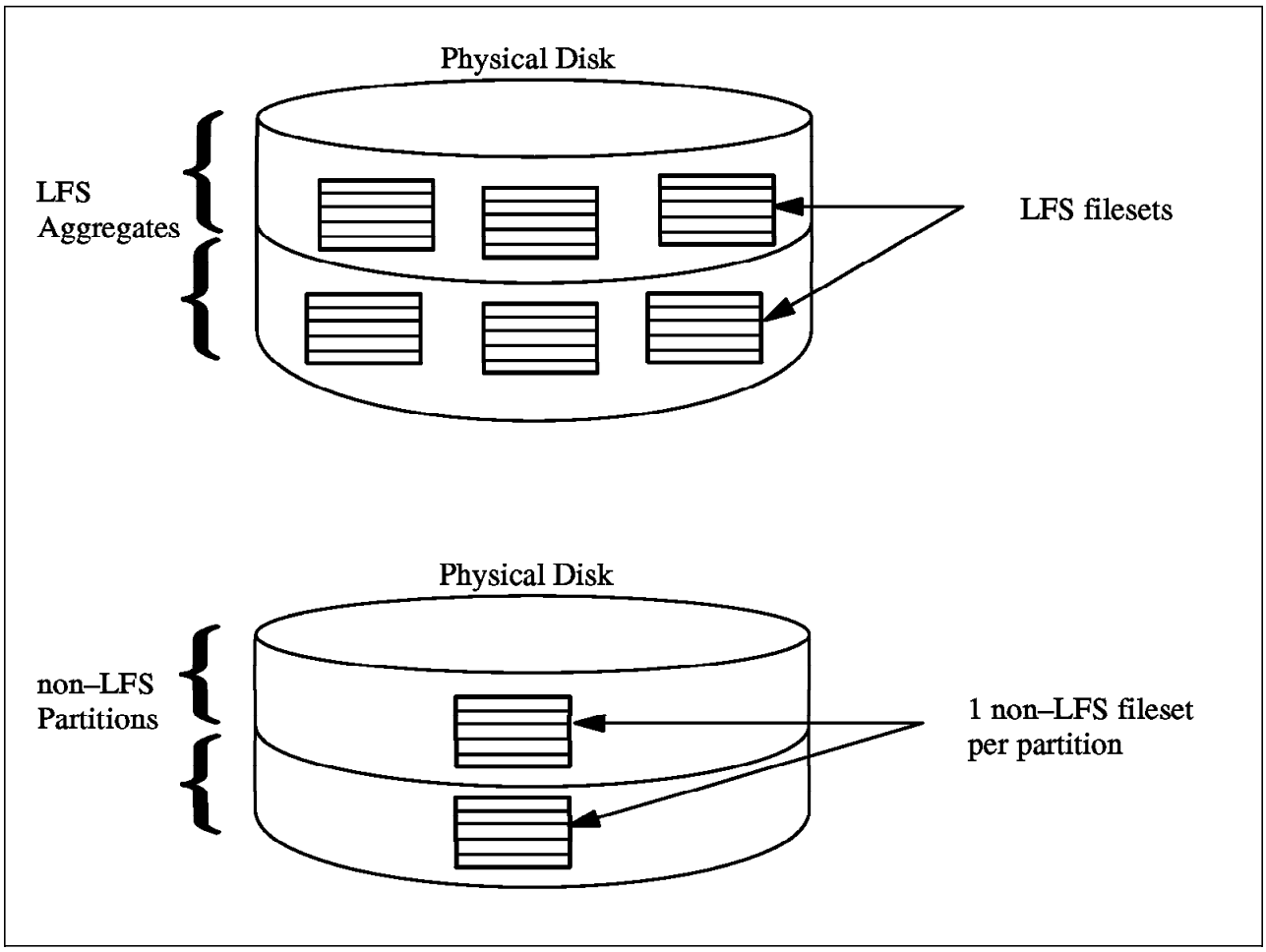


Figure 28. DCE LFS and Non-LFS Disk partitioning

3.3.2 Advantages of the DCE LFS-Based File System

3.3.2.1 Replication With DCE LFS Filesets

The organization of DCE LFS filesets into aggregates provides a number of features not found on ordinary file systems. First, the concept of DCE filesets as logical subsets of aggregates means that filesets are not tied to a physical location on the storage media and can therefore be manipulated independently. This allows filesets to be moved among partitions or moved from one server to another.

The Replication Server is an administrative server that handles replication of filesets. For example, an administrator can create a copy of a fileset on a second File Server machine. The Replication Server will update the replica at a specified interval (every 30 minutes, for example). This means that even if the file′s master File Server machine goes down, a copy of the file is still available online through the secondary copy on the alternate File Server machine. The period of time that the replicas are updated is defined by the administrator when setting up the replication for the fileset.

Note

Replication only operates on DCE LFS filesets. It cannot be applied to non-LFS (JFS) filesets. Replication is not supported in the AIX DCE Enhanced Distributed File System/6000 Version 1.1 release but will be supported in a future release.

More details about the Replication process will be explained later in section 3.6, “DFS Replication - What It Is and How It Works” on page 67.

3.3.2.2 DCE LFS Clones & Backup Filesets

A snapshot of a fileset (called a clone) can be created on the same aggregate as the Read/Write fileset. DFS Clones and DFS Backup filesets are equivalent terms for the same subject. Unfortunately the term backup is used also in conjunction with the DFS Backup System, which is explained in 3.5, “ Backing Up DFS Filesets to Tape” on page 61 and which has primarily nothing to do with clone or backup filesets.

As mentioned before, there are three different types of LFS filesets. They are:

1. the readWrite version
2. the readOnly version
3. the backup version

There will always be only one readWrite version of a fileset. There can be zero or one backup version and there can be zero or several readOnly versions of the fileset.

The backup version is not really a copy but a clone which uses pointers to the same physical data blocks as the readWrite version. Therefore it uses much less disk space. In fact, it uses no disk space for data blocks, just the space for fileset header information and the space for the file inodes which contain pointers to the data blocks.

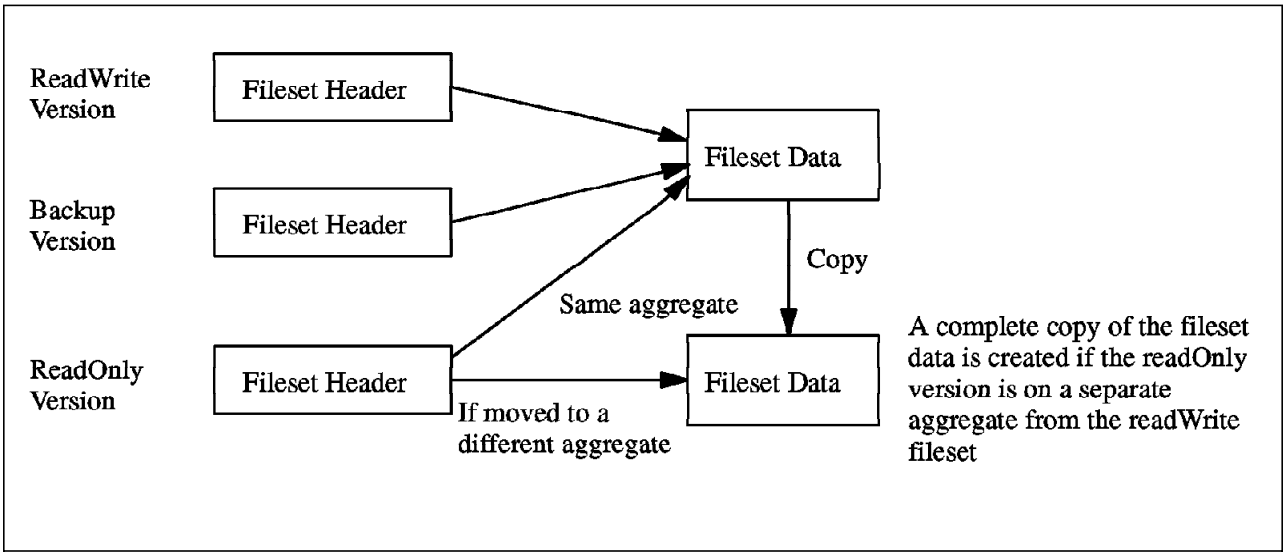


Figure 29. How Fileset Types Relate to Fileset Data

Since the backup version of the fileset simply uses pointers to the data blocks of the readWrite version, it must always be stored at the same site as the readWrite version. (The readOnly version can be copied to remote sites. This is known as replication.) The backup version is a *snapshot* of the readWrite version. It is used to make data available for other tasks, such as the DFS Backup System. The purpose of the readOnly version is to provide ways of making the data available on multiple sites for faster access and higher availability.

The backup version is an exact clone of the readWrite version at the time when the backup was created. The data blocks are not copied but both filesets will have pointers set to the data blocks. If the readWrite version updates its data, new data blocks are created for the readWrite fileset and pointers to this data for the readWrite fileset are changed accordingly. This is shown in Figure 30 on page 53.

The readOnly version of a fileset will exist on the same site as the readWrite version of the fileset when scheduled replication is used. When the readOnly version exists on the same site as the readWrite version, a clone (backup) fileset is actually used. Therefore all the data blocks are not copied but inode pointers are used to point to the data blocks. If this readOnly fileset is copied to another aggregate, then the data blocks will also be copied. A readOnly fileset on a remote site will also have this copying of the data blocks when the readOnly fileset is created on that remote site.

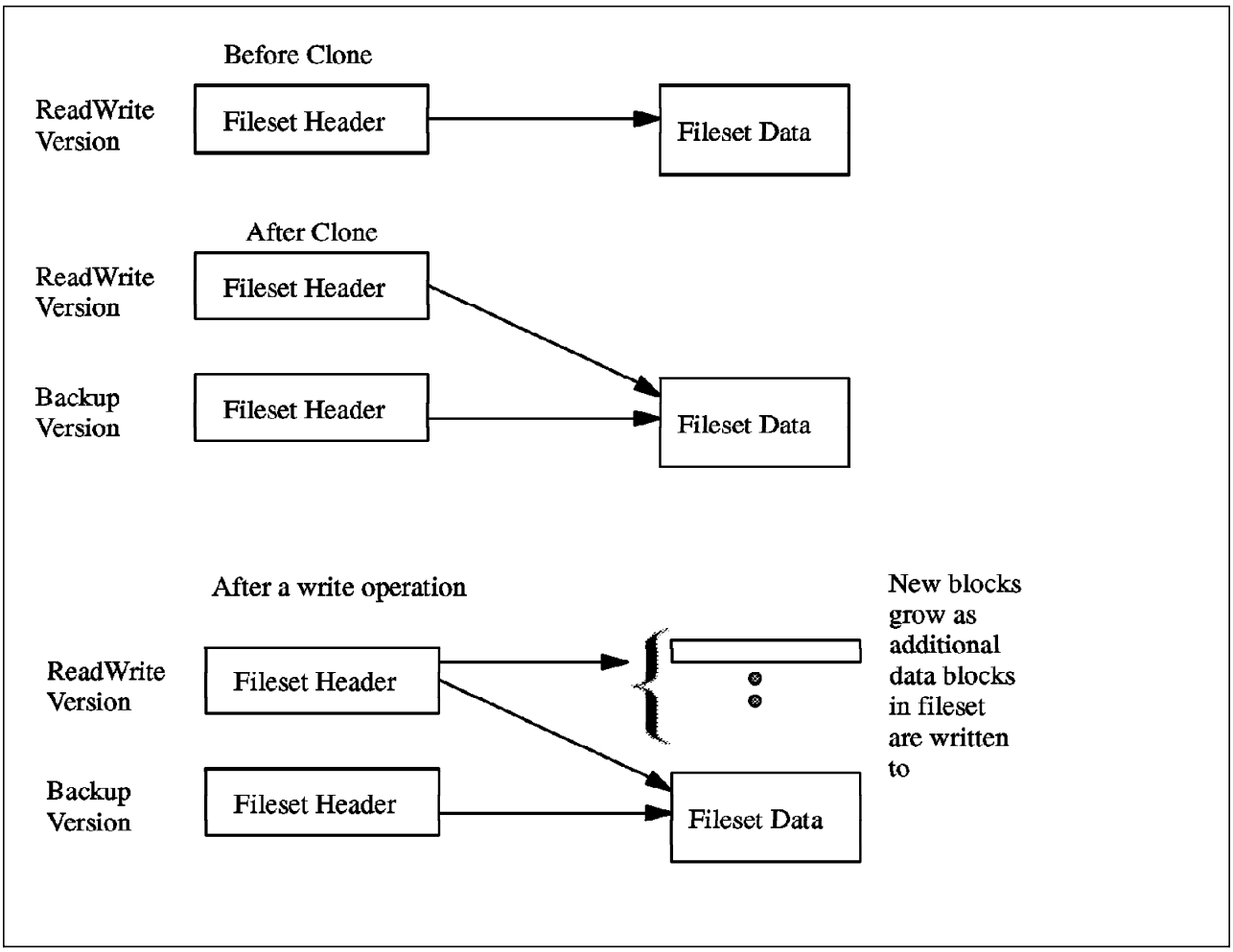


Figure 30. Fileset Data and Fileset Type Relationship When Cloning Occurs

3.3.2.3 The DCE LFS Log-Based File System

Data available in filesets and aggregates can be divided into two general types: *user data* and *metadata*. User data is the data in a fileset (applications and data files) that is created and referenced by users.

Metadata is the data in any part of a file system that is used to describe and organize the files and directories in that fileset or aggregate. Metadata is logged in DCE LFS but user data is not.

Each aggregate has its own log. The size of the log for each aggregate is fixed when that aggregate is initialized. When the amount of data held in the log approaches the limits of the space allocated to the log for that aggregate, the processing of new file system requests is halted to allow pending transactions to commit. File system changes are then flushed to disk, insuring that the log space previously required for records of those operations can be freed.

DCE LFS only logs file system *metadata*, such as ACLs, directory entries, protection modes and file and directory status information. It does not keep a log of the changes to the user data in a partition. Maintaining a log of changes to user data would require larger, longer transactions that could easily require gigabytes of storage for log records.

In conventional UNIX file systems, system failures that interrupt certain operations can leave the file system in an inconsistent state, making it potentially unsafe to use and requiring the use of the fsck command after reboot to check for and repair any damage.

Maintaining a record of the changes made to metadata enables the DCE LFS to resume normal operation quickly after a system failure. After a system crash, DCE LFS applies recovery techniques that either undo operations that have begun but not finished, or complete operations that have finished but not yet been written to the file system. The time spent in recovery is proportional to the size of the active portion of the log and not to the size of the file system (as with fsck).

3.3.3 Pathnames, Mount Points and Local Mounting

Files and directories in DFS are accessed by specifying DFS pathnames. The filesets are mounted at DFS mount points. This section explains the concept of pathnames as well as mount points.

3.3.3.1 AIX Pathnames and DCE DFS Pathnames

With AIX and the DFS, you can refer to files and directories by both AIX pathnames and DFS pathnames. The AIX pathname is the same as a UNIX pathname. It is used to locate files and directories in the file tree structure. DFS has a special syntax to identify the files and directories in the DFS name space.

A quick review of DCE pathnames is as follows: In DCE you would use */.../<cellname>/fs* to get to the root directory of a DFS file tree structure.

This can also be written as */.:/fs* and it will point you to the root directory of the DFS file system of the cell that the DFS client is defined in.

You can also shorten this to */:*

Below are some examples of AIX and of DFS pathnames. Note that the DFS pathnames all refer to the same file called myfile.

Examples of pathnames:

1. AIX pathnames
   - /home/user1/file1
   - ./file2
   - ../user1/file3
2. DFS pathnames
   - /.../dino/fs/myfile
   - /.:/fs/myfile
   - /:/myfile

Recommendation

It is recommended for shell programming that a fully qualified pathname (/.../cellname) should be used instead of : (colon). This is because some shells tend to interpet the colon character.

3.3.3.2 Mount Points

The mount point is the file system object that we want to attach a file system to so that we can seamlessly access files and directories in our newly mounted file system. In the AIX file system, the mount point is a directory where it is permitted to attach a file system. This is possible through the `mount` command. When this directory is accessed as part of the path name to the file or directory inside the file system, the operating system recognizes it as a *bridge* to the device that contains the data organized.

Inside DCE DFS we have the same concept of a mount point. Instead of using a directory to attach the file system (in this context, fileset), DCE DFS defines special objects that are created, manipulated and deleted using DCE DFS commands. These objects are created using the `fts crmount` command.

In the AIX file system and the DCE DFS filesets, once the mount point is created and the device is attached, the user or the application program does not notice the difference between an ordinary directory and the mount point.

Examples:

1. AIX mount points - These are automatically created by installing AIX

Table 2. AIX Mount Points and Associated Devices

Mount Point	**File System Located on Logical Volume**
/	/dev/hd4
/var	/dev/hd9var
/usr	/dev/hd2
/tmp	/dev/hd3
/home	/dev/hd1

2. DCE DFS Mount Points - These were created specifically for this example.

Table 3. DCE DFS Mount Points and Associated Fileset

Mount Point	**Fileset**
/:/user/user1	user.user1
/:/user/user1.backup	user.user1.backup
/:/user/user2	user.user2
/:/user/user2.backup	user.user2.backup
/:/user/user3	user.user3
/:/user/user3.backup	user.user3.backup
/:/user/user4	user.user4
/:/user/user4.backup	user.user4.backup

Note that by using the `fts lsmount` command, we can list whether a specified directory serves as a mount point for a fileset.

```
# fts lsmount -dir /:/user/user1
'/:/user/user1' is a mount point for fileset '#user.user1'
```

3.3.3.3 Types of Mount Points

With DCE DFS you have many different ways to access data inside a fileset. You have the read-write fileset, the corresponding backup fileset and possibly read-only replicas. For each of these filesets it is possible to have mount points and have the data visible. However, the operations that you can perform with the data is different.

The mount point is characterized by its:

- fileset type (read-write, read-only, or backup)
- mount point type (regular or read-write)
- location type (local)

The *fileset type* indicates which type of fileset is pointed to by the mount point. It can point to the read-write, read-only (extension *.readonly*), or backup (extension *.backup*) filesets.

A read-write fileset is just what it implies. It is a fileset that can be used for reading and/or writing files and directories.

A readOnly version of a fileset is a copy of the readWrite version of an LFS fileset. The readOnly fileset uses as much disk space as the source. There can be several versions of the readOnly copy of the fileset whenever you desire files to be highly available. You can only read and execute files from a readOnly fileset.

A backup version of a fileset is a clone which uses pointers to the same physical data blocks as the readWrite version. Therefore it uses much less disk space. You can only read and execute files from a backup fileset.

The *mount point type* can either be regular or read-write. A regular mount point is a type of a mount point that allows a cache manager to select the type of file system (ReadOnly, ReadWrite or Backup) according to the algorithm in Figure 31 on page 57. A ReadWrite mount point directs the cache manager to always use the ReadWrite file system.

Mount points are usually of the *regular* type. The *read-write* mount point type is used to point to the read-write fileset only, regardless where the mount point resides.

Mount point types are important because of how the cache manager on a DFS client resolves a file pathname. The cache manager resolves a file pathname by starting at the top level fileset in the cell (root.dfs) and accessing the readOnly version filesets (whenever possible). Figure 31 on page 57 describes the algorithm that a cache manager uses to traverse the mounted file systems to find the file. This topic is covered in more detail in 3.6, “DFS Replication - What It Is and How It Works” on page 67.

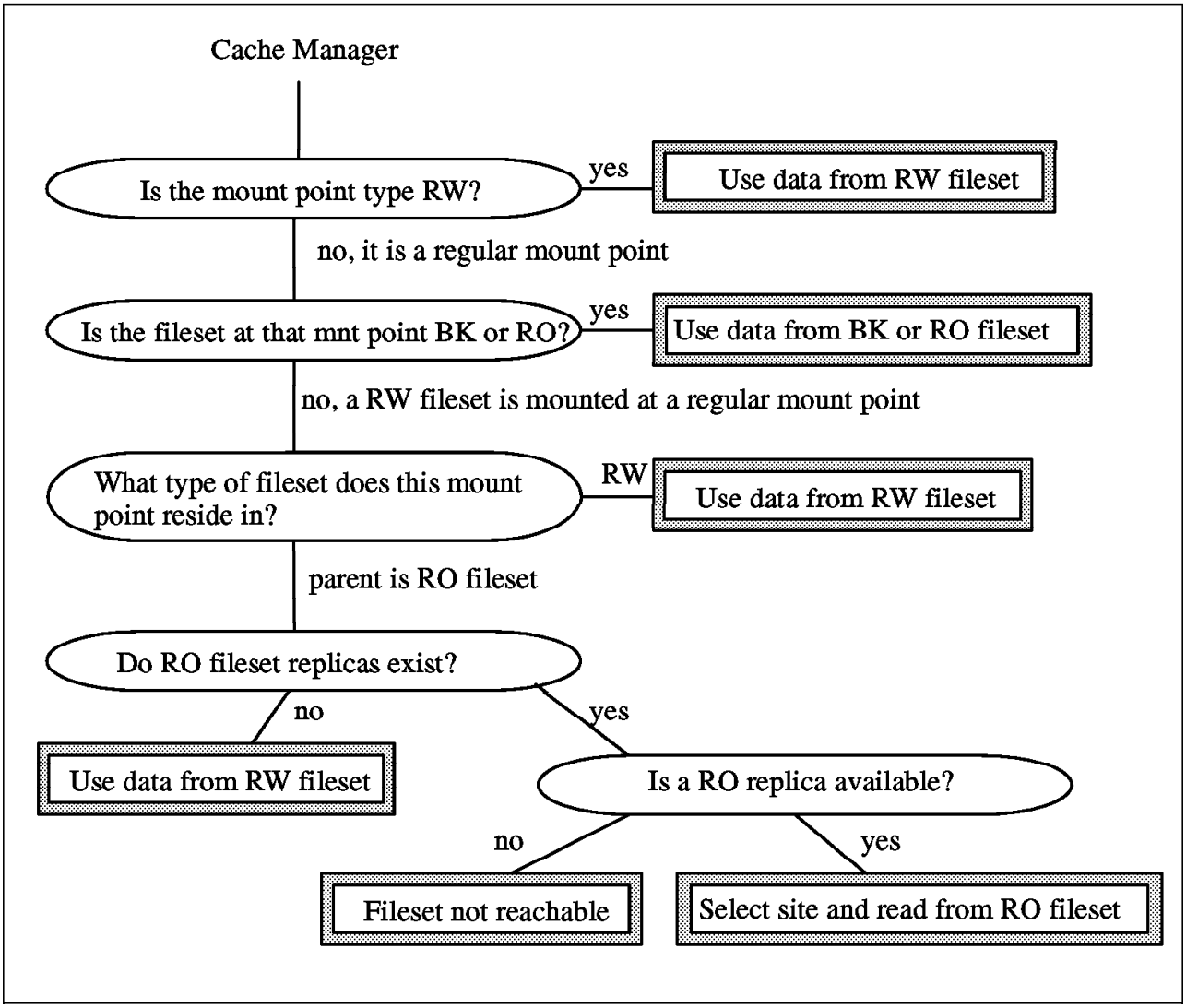


Figure 31. Decision Path of a Cache Manager Selecting a Fileset

The following table summarizes the relationship between mount point types, the fileset types they point to and the fileset types they reside in, indicating the fileset the Cache Manager will then access.

Table 4 (Page 1 of 2). Mount Point Types, Fileset Types and Fileset Access

Mount Point Type	Fileset Pointed To	Fileset Resided In	Fileset Accessed
regular	read-only	any type	read-only
regular	backup	any type	backup
regular	read-write	read-write	ONLY read-write
regular	read-write	read-only	tries access to read-only • if no read-only defined, access read-write • if read-only defined but unavailable, fileset unreachable

Table 4 (Page 2 of 2). Mount Point Types, Fileset Types and Fileset Access			
Mount Point Type	**Fileset Pointed To**	**Fileset Resided In**	**Fileset Accessed**
read-write	ONLY read-write	any type	ONLY read-write

Using fts lsmount command you can examine DFS mount points. You can see their types and the fileset associated. The output of the command has the following format:

```
# fts lsmount -dir /:/user/user1
'/:/user/user1' is a mount point for fileset '#user.user1'
```

The fileset name also indicates the mount point type, with the signs:

- # (pound sign): regular mount point;
- % (percent sign): read-write mount point;

3.3.3.4 Local Mounting

DCE DFS also allows you to access objects inside a fileset using AIX pathnames. This is the concept of a *local mount point*. A local mount point is a directory over which you can mount a DCE DFS fileset. In this case, you can access DCE DFS objects inside the fileset without the special syntax that defines the DCE pathname. It will look like the pathname refers to an ordinary AIX object but actually you will access a DCE DFS object.

To create the local mount point you use the regular AIX mount command with different arguments:

mount -v lfs -o aggregate=<aggregatename> **-n** <hostname> <filesetname> <directory on which to mount>

Examples:

```
# fts lsmount -dir /:/user/user1
'/:/user/user1' is a mount point for fileset '#user.user1'
# mount -v lfs -o aggregate=user1_lv1 -n `hostname` user.user1 /mnt/user
# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
...
         /dev/user1_lv1   /mnt/user        lfs    Jul 23 14:58  user.user1
...
```

In this case, you can access a file named *filename* in a DCE DFS directory as */:/user/user1/filename* and the same file will be accessible using AIX pathname */mnt/user/filename*

Limitations of Locally Mounting DCE LFS Filesets

- Local mount points are visible only for the users of the local File Server Machine. Users from other machines are not able to locate the objects in the local mounted file tree.
- Locally mounted DCE LFS file systems cannot be replicated.
- Permissions used to access the locally mounted filesets are the AIX mode bit permissions for your AIX logon ID. DCE ACLs are not used.

It is recommended that you only use local mounting of DCE LFS filesets to perform emergency system administration.

3.4 Accessing DFS Data

This section will describe the concepts concerned about Cache Manager, File Exporter, Token Management, Replication and Clones. They deal with the management of data and access,how they work and the benefits of these features.

3.4.1 Cache Manager

The Cache Manager runs on any DFS client machine. It takes a user′s file system request and looks in a cache to see if a copy of the data is already on the local system. If not, the Cache Manager sends a request to the File Server machine for the data and then caches it locally. Since files are cached on the client, a local copy of the file can subsequently be accessed instead of the remote copy on the File Server machine. As a result, network traffic to the File Server machine, as well as File Server machine load, is much lighter than if the client always had to go to the server each time it needed to access the file.

The DFS Cache Manager is responsible for the local caching of file and directory data on machines used as file system clients. *Caching* means that once any portion of a file is requested and retrieved by a client, a copy of that data is kept on the client machine. There are tokens associated with the cached data. These tokens help the file server synchronize the file data so that the user is always working with the most up to date version of the file.

3.4.2 File Exporter

The File Exporter runs on a DFS File Server machine. It handles requests from remote clients for the files that it manages. The File Exporter receives an RPC call and accesses its own local file system. The file system which can be the DCE Local File System (LFS) or another file system such as the AIX journaled file system (JFS) services the request. Using the Token Manager, it handles the synchronization of different clients which may be concurrently accessing the same file. It then returns the requested information to the client.

The *Token Manager*, called by the File Exporter, maintains the set of file and directory tokens that have been granted to existing clients of that File Server machine. These tokens have a defined compatibility relationship that specifies the other token types that may be simultaneously granted to other clients.

3.4.3 Tokens and the File Exporter

In a distributed environment it is very common to have one object being accessed by different users who can be logged in from different systems. They may want to simultaneously modify these objects. DCE DFS includes the Token Manager in the File Server kernel to guarantee synchronization to the objects.

There is one Token Manager for each File Server and it keeps track of objects stored on this File Server machine. For each operation involving file server objects, one or more tokens will be sent to the client. This is a guarantee to the client that it has the permissions that it requested.

This mechanism is used to enforce a coherent view of the data across all DFS clients. A client in this case is any local or remote application that requests access to the object. If conflicts occur, the Token Manager attempts to revoke the conflicting tokens to satisfy the requests.

The File Exporter of the File Server manages the tokens. The token is actually a data structure passed from the server to the client.

The token mechanism is not a way to provide automatic file locking. The Token Manager supports byte range file locking but the application must manage its file locks in the same way it does in other environments. Tokens are used by DCE DFS to synchronize access to data and maintain cache consistency between the File Server and its clients. It is transparent to the application.

3.4.4 Token State Recovery (TSR)

The Token Management Mechanism relies on the two-way communication between the File Server (File Exporter) and its clients. Sometimes this communication is broken because the server/client is down or because there is a network problem. In these cases, some recovery is made possible by the File Exporter.

This recovery is necessary to give the clients an opportunity to rebuild the information of its tokens in the server and prevent the server to grant conflicting tokens after a crash/restart.

In the case of the File Server machine is restarted, the File Exporter enters in token recovery state and attempts to preserve the state of its tokens by initially accepting only requests to reestablish existing tokens from the clients. During the time that the server was out, the clients still attempt to contact it to prove that they are alive. By these contacts, it is possible to recover the state of the tokens after the server is running again. The amount of time spent in TSR mode is influenced by the parameters passed to the fxd daemon. See IBM InfoExplorer for more detail.

Eliminating the Token Recovery Time

At each restart of the File Server, it waits a amount of time for the clients to reestablish the state of its tokens. This happens even if there are no clients to wait for. It makes the data exported by the File Server unavailable during this time. You can set the token start recovery time to zero by changing the fxd line in the /etc/rc.dfs file to:

```
daemonrunning $DCELOCAL/bin/fxd -mainprocs 7 -admingroup   \
subsys/dce/dfs-admin -notsr
```

The -notsr parameter disables the token state recovery. Note that by doing this, you might lose the ability to have the previously connected DFS clients recover file server contact using the existing tokens that they are holding. This may result in the DFS clients not being able to restore their changed data back to the file server. This will also affect DFS client applications. (For example, a page fault that occurs when the DFS client machine is executing a binary could possibly result in a core dump.) This is not recommended for production environments.

If the client machine is unable to contact the server, it will not be able to prevent the expiration of the Token Lifetime of its tokens. After the communication is reestablished, the client will need to request new tokens.

In the case of the client machine being restarted, no information about its tokens is available and there is no recovery possible.

3.5 Backing Up DFS Filesets to Tape

There are several ways to backup filesets. There is the DFS Backup System method and the standard UNIX backup methods such as tar, dd or cpio. This chapter describes the DFS Backup System method. Additional information about the administrator′s usage of the DFS Backup System can be found in Chapter 5, “DFS Administration Tasks” on page 109.

The DFS Backup System provides procedures to automate the backup of filesets (DCE LFS and non-LFS) to tape and to restore the data at appropriate times back to the fileset. Information about how to backup, which tapes to use and when to backup is maintained in the Backup Database. The Backup Database can also be written to tape and restored which is necessary in the event of a complete system corruption.

The following figure shows the system sys1 being configured as the DFS Backup System. In our example, the system sys1 also acts as the Tape Coordinator. This makes it necessary that this machine have at least one physical tape unit attached. In general, the Tape Coordinator machine can have several tape units attached. The DFS Backup System machine together with the Backup Database has access to all filesets in the DFS administrative domain and holds policies when to backup these filesets.

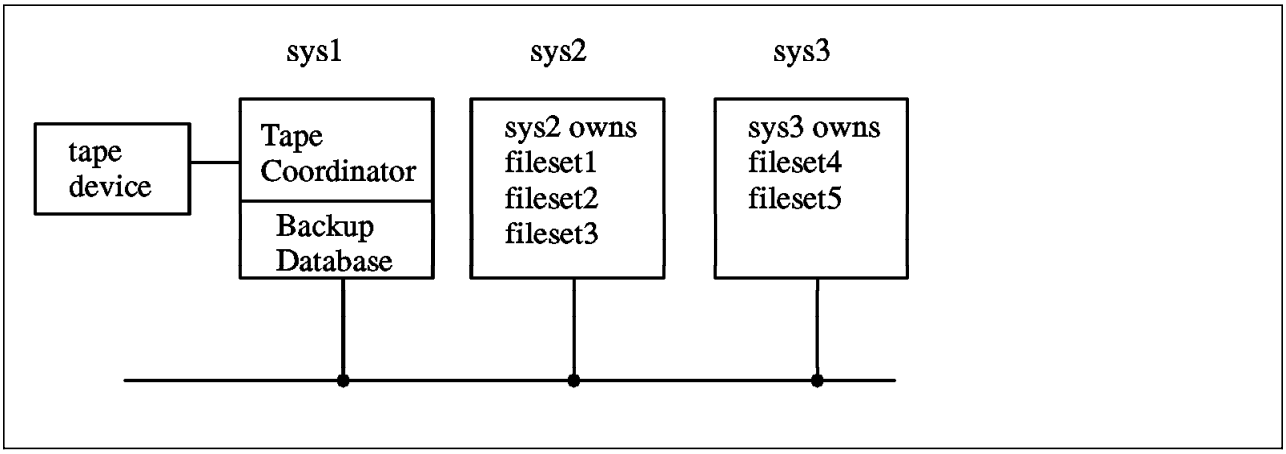


Figure 32. Overview of DFS Backup Machine

Figure 32 demonstrates the machine role of the DFS Backup System sys1. This machine acts also as Tape Coordinator. It is recommended to combine these two machine roles but it is not necessary.

Backups can be made as full backups or incremental backups. Full backups contain all data in a fileset. Incremental backups hold the data that has changed from the last incremental or full backup. Incremental backups have a parent pointer to either a full backup or another incremental backup. In the case of restoring a fileset, you have to work backwards starting with the oldest parent

(this is always a full backup) and then restoring all incremental backups in the same sequence as they were taken during the backup procedure. We recommend that you maintain full backups in a periodic time schedule, such as on a weekly or monthly base and provide incremental backups in between the full backups. Of course, it would be easier to always take full backups but the incremental method minimizes the time and effort to take backups. It also provides enough flexibility for a secure system backup strategy.

The DFS Backup System method also provides ways to group filesets in families. The purpose of this is to allow the system administrator to keep those filesets together that have some kind of relationship and to dump or restore these as an entity.

The DFS Backup System allows you to mix data from DCE LFS filesets and non-LFS filesets. It can also be used to dump data from an DCE LFS fileset and to restore this data to a non-LFS fileset. Likewise, you can dump data from a non-LFS fileset and restore it to an DCE LFS fileset. However, be aware that non-LFS filesets cannot use ACLs. Therefore you will lose the information carried in ACLs if you dump from DCE LFS filesets and restore to non-LFS filesets.

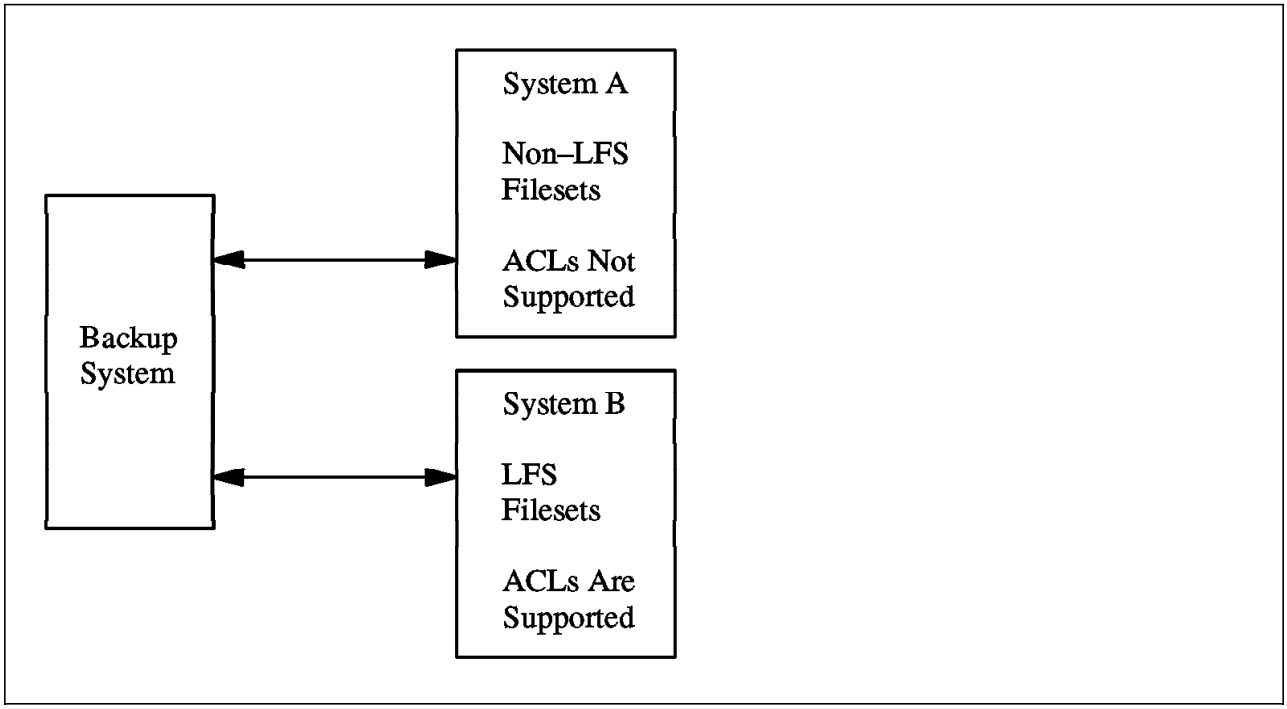


Figure 33. Dumping and Restoring of DCE LFS and non-LFS Filesets

3.5.1 Backup System Database

The Backup System Database holds information about schedules, locations, families and other administrative data. There should be only one master Backup Database per cell that is used for all DFS administrative domains in a cell. To ensure the availability of the Backup Database you can replicate the Backup Database. We do recommend that you start with only one Backup Database and add the other backup databases when you need the higher availability. In addition, you should also backup the database itself by dumping it to tape so it can be restored in case of loss. There will not be too many updates to the database since the administrative schedule will typically not change frequently. It

may be sufficient to backup the Database occasionally. Note that the Backup System Databases operate in the same manner as the FLDBs with regard to voting, quorum, and so forth.

The Backup Database is located in the /var/dce/dfs/backup directory. The database is in a binary format and you cannot view or edit the data with any of the UNIX standard commands such as vi or pg. Instead use the bak command suite from DFS to alter or list the contents of the database. Only the administrative users and members of the groups which are included in the admin.bak administrative list can issue bak commands.

3.5.2 Tape Coordinator

The DFS Tape Coordinator Machine is the system which physically writes and reads the data to the tape. Therefore the DFS Tape Coordinator Machine must have at least one tape unit attached. The DFS Tape Coordinator Machine must also be configured as such with the bak addhost command and it must run one or more processes called *butc*. You want to run one *butc* process for every tape unit which is physically attached and under control of the DFS Tape Coordinator Machine.

A Tape Coordinator has an identification number called a tape coordinator ID or TCID. When you issue bak commands you may use the TCID, which is a unique number in the cell to identify the tape coordinator. If you have several tapes attached to the DFS Tape Coordinator Machine, you will have one tape coordinator process (and unique TCID) for each tape on that machine. The tape coordinators are listed in a configuration file located at /var/dce/dfs/backup/TapeConfig on the DFS Tape Coordinator Machine. Here is an example of the configuration file:

```
# pg /var/dce/dfs/backup/TapeConfig
1g 4k /dev/rmt0  0
2g 200k /dev/rmt1  1
1g 4k /dev/rmt2  2
2g 1m /dev/rmt3  3
```

Entries in the tape coordinator configuration file have four fields. The first field is the overall tape size. The Tape Coordinator uses this capacity whenever a tape is used in the drive. The second field defines the size of End_Of_File (EOF) tape marks. One of these tape marks is written after each fileset dump and the size of the tape mark can affect the amount of space available for data. It is not unusual to have tape marks of up to 2MB. The third field specifies the physical tape unit and the fourth field is the TCID (Tape Coordinator ID).

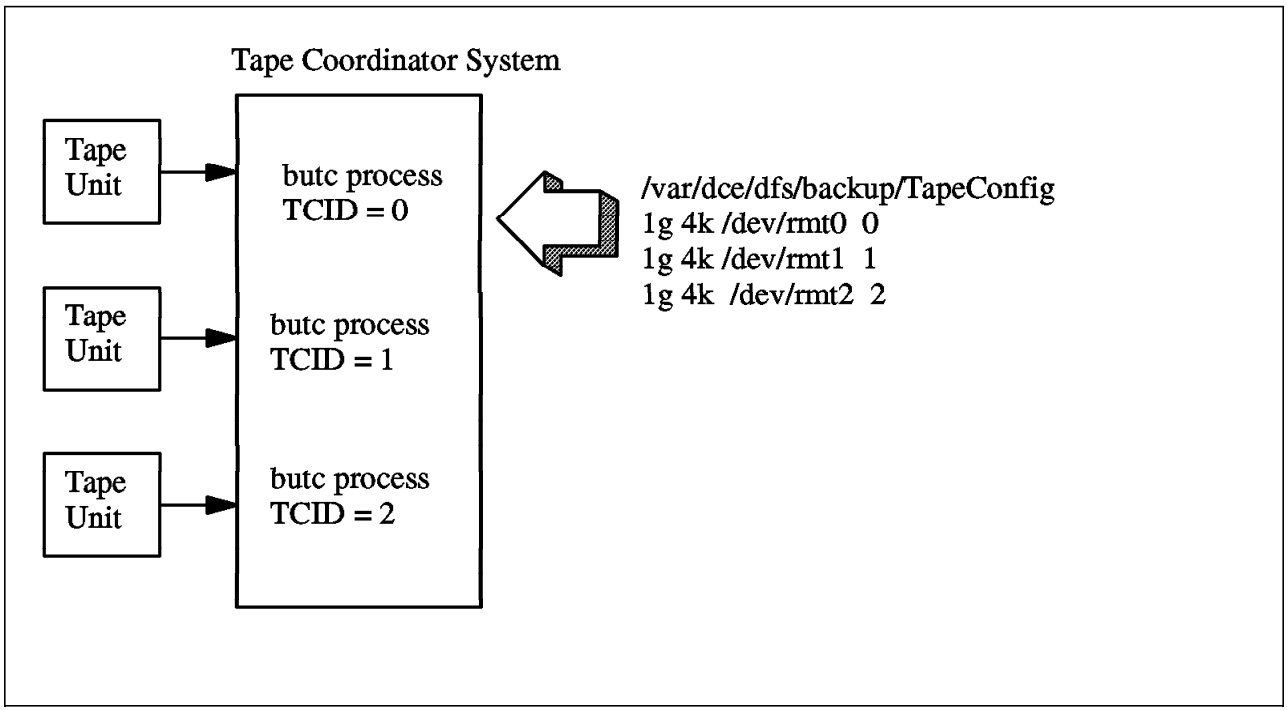


Figure 34. Tape Coordinator Machine

Figure 34 shows the DFS Tape Coordinator Machine with three tape coordinators. Each has an entry in the configuration file. Tape Coordinators are started with the simple command "butc" and they run in the foreground. This means that these sessions can not be used for any other activity. To handle several Tape Coordinators, it is best to open a separate window for each Tape Coordinator under X-Windows** and keep this window active (iconized or restored) as long as the tape is in use with the DFS Backup System. To stop a Tape Coordinator enter the <Ctrl>-c key combination in the specified window.

3.5.3 Fileset Families

To ease the handling of many filesets spread all over your administrative domain, you can group filesets into families of filesets. The filesets of a family have usually some kind of relationship. For example, if they are all user data of a specific group of users it may make sense to dump or restore these filesets together on the same tape. The filesets which belong to the same family are bound to the same schedule and to the same tape set. If you have set up the family to be dumped as a full dump once a week, then all of the filesets in that family will be dumped once a week.

The entries in fileset families have three fields. Look at the following example:

```
server /.../dino/hosts/sys3, aggregate lfs.sys3 filesets: user3.*
server /.../dino/hosts/sys3, aggregate lfs.niki filesets: user3.niki
server /.../dino/hosts/sys1, aggregate user1_lv filesets: sys1.*
```

The first field defines the file server machine where the fileset resides. The second field specifies the aggregate and the third field names the fileset. This information has to match with the Fileset Database, which is stored on your Fileset Location Database Machine. Note that * can be used as a wild card. Therefore user3.* is used to define all filesets that start with the "user3." prefix.

3.5.4 Dump Levels

Dump levels define a dump hierarchy. This is a logical structure that defines the relationship between full dumps and sequences of incremental dumps. Since incremental dumps only hold the changes since the last dump, they require a parent dump to which they have set up a pointer. Eventually, incremental dumps point back to a full dump although there may be several levels of incremental dumps. The Backup System keeps all the data that belong to a dump set together on a tape or a set of tapes.

Let us look at some examples of dump hierarchies which are based on a weekly schedule. Figure 35 shows a daily dump, where full dumps are taken on Sunday and incremental dumps are taken on weekdays. Each weekday dump is an incremental dump to the previous day's dump. If you encounter a need to restore on Thursday, you must restore the full dump from last Sunday then Monday's dump, and Tuesday's dump and Wednesday's dump in that exact sequence.

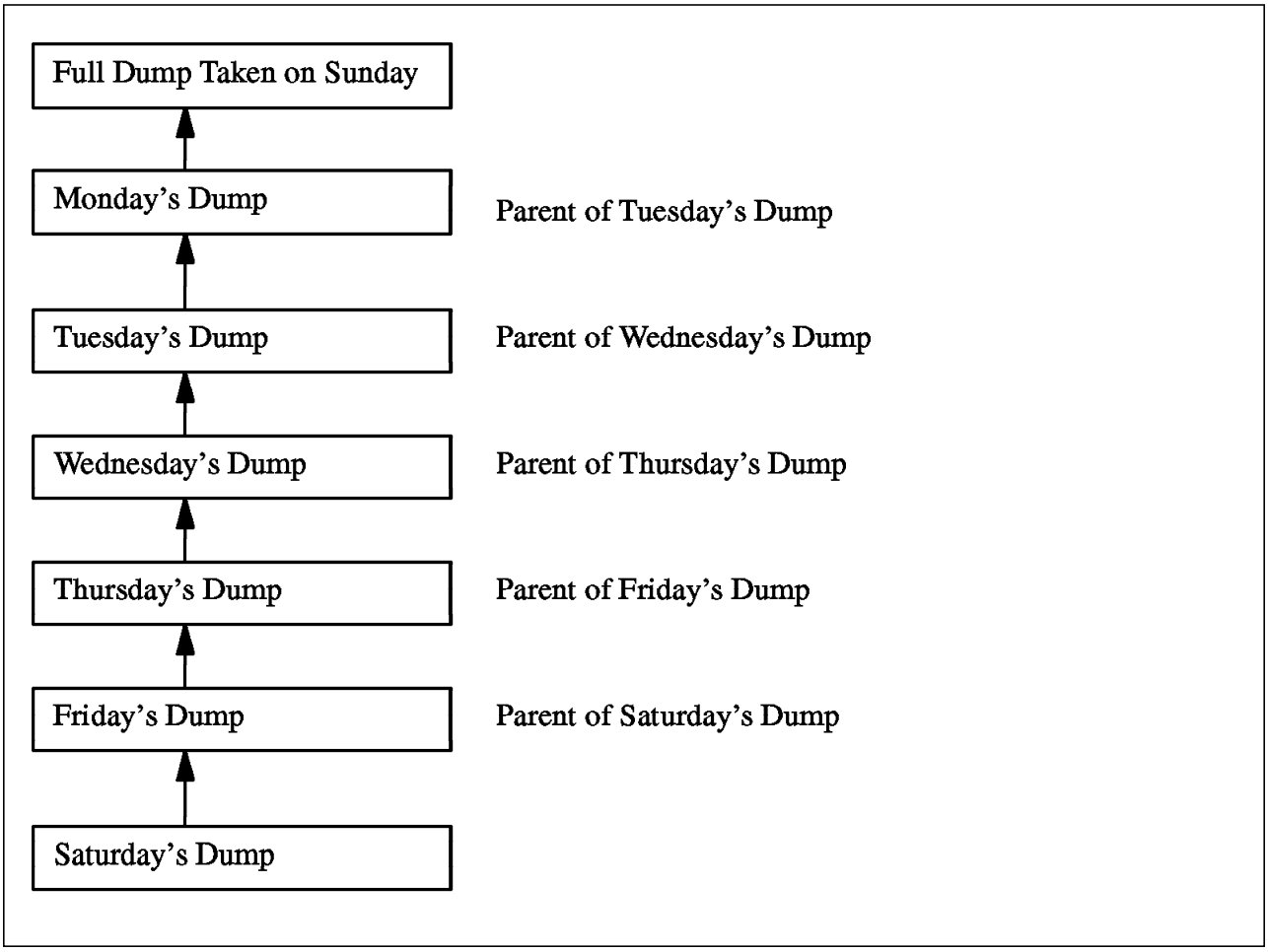


Figure 35. Example of Dump Hierarchy

In case of restoring a fileset close to the end of the week you must restore the dumps of all previous days of the week. The corresponding commands to enter this dump hierarchy into the Backup Database are:

```
# bak adddump -l /sun -expires in 1w
# bak adddump -l /sun/mon -expires in 1w
# bak adddump -l /sun/mon/tue -expires in 1w
# bak adddump -l /sun/mon/tue/wed -expires in 1w
# bak adddump -l /sun/mon/tue/wed/thu -expires in 1w
# bak adddump -l /sun/mon/tue/wed/thu/fri -expires in 1w
# bak adddump -l /sun/mon/tue/wed/thu/fri/sat -expires in 1w
```

Figure 36 shows a dump hierarchy where each weekday dump depends only on Sunday's dump. This method will use more data but it will require less time to restore. It also is more reliable since it only depends on Sunday's dump and not on all previous weekday dumps.

Note that the expiration date is assigned by the administrator so that the administrator cannot use the `bak dump` command to inadvertently write over the data on the tape.

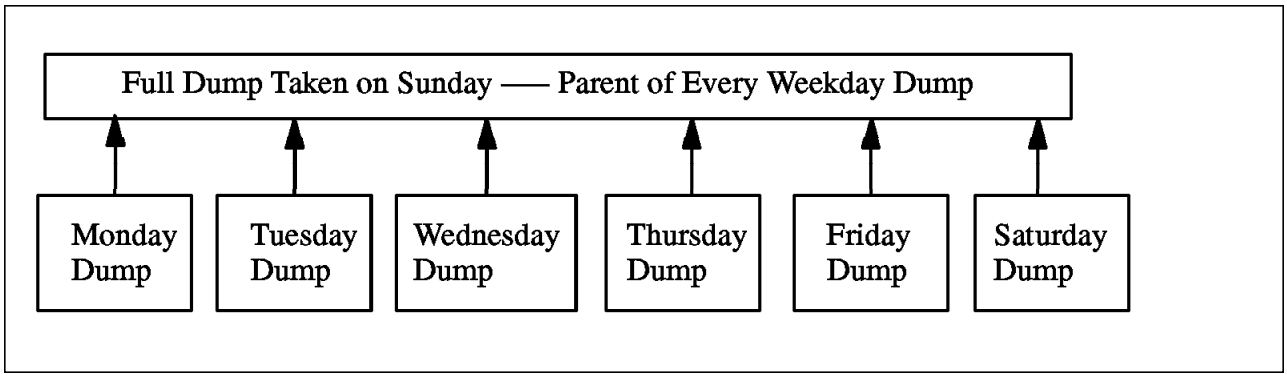


Figure 36. Only A Parent Dump and One Incremental Dump Need be Restored

The corresponding commands to enter this hierarchy into the Backup Database are:

```
# bak adddump -l /sun -expires in 1w
# bak adddump -l /sun/mon -expires in 1d
# bak adddump -l /sun/tue -expires in 1d
# bak adddump -l /sun/wed -expires in 1d
# bak adddump -l /sun/thu -expires in 1d
# bak adddump -l /sun/thu -expires in 1d
# bak adddump -l /sun/fri -expires in 1d
# bak adddump -l /sun/sat -expires in 1d
```

3.5.5 Privileges Required for Backup System

Administrators who work with the DFS Backup System must be listed in each of the following administrative lists:

1. admin.ft
2. admin.bak
3. admin.fl

Be aware that these lists are distributed by the DFS System Control Machine. If you entered the names locally then these might be overwritten by the upclient process. Therefore, make the changes on the DFS System Control Machine. As an example, assume the administrator of the Backup System belongs to the group *bakadm*. You must define this group in all three administration lists as follows:

```
# bos addadmin -s <hostname_of_SCM> -adminlist admin.ft -group bakadm
# bos addadmin -s <hostname_of_SCM> -adminlist admin.fl -group bakadm
# bos addadmin -s <hostname_of_SCM> -adminlist admin.bak -group bakadm
```

The default setup allows the principal cell_admin to perform the commands for the DFS Backup Machine

You can verify the existence of the correct entries in these lists with:

```
# bos lsadmin -s <hostname_of_SCM> -adminlist admin.ft
# bos lsadmin -s <hostname_of_SCM> -adminlist admin.fl
# bos lsadmin -s <hostname_of_SCM> -adminlist admin.bak
```

3.5.6 Tape Labels

The DFS Backup System maintains tape labels on those tapes which are in use with the Tape Coordinator. The labels hold information about the tape such as size and tape marks. They also hold information about the data it contains. Using tape labels guarantees that you do not accidentally write to a wrong tape and overwrite data which you may want to keep. The name of the tape label is composed by the system and has the following three fields separated with a period:

1. Family_name
2. Dump_level
3. Index

The Family_name and the dump_level name must match whatever you have defined as fileset families and dump levels. The index indicates the place in a sequence of tapes if more than one tape is required to dump a family set.

The DFS Backup System will check these labels for every write operation to a tape and will refuse tapes with the wrong label or with an unexpired date. It is not necessary to pre-label all tapes, but the system will write labels once the tape is used. If you need to overwrite a tape with an unexpired date, you may use the `tctl` AIX command to destroy the label and write a new one.

```
# tctl -f /dev/rmt0 rewind
# tctl -f /dev/rmt0 weof 1
# tctl -f /dev/rmt0 rewind
# bak tapelabel
```

This will rewind the tape, write one tape mark at the beginning of the tape and therefore destroy the label. Then it will write an empty label at the beginning of the tape so the tape is usable for further dumps.

3.6 DFS Replication - What It Is and How It Works

Replication is the process of creating one or more read-only copies of the complete read-write DCE LFS fileset. This allows for ReadOnly copies on multiple sites. Therefore if you have one machine failure you can access the information from a different site.

Read-only filesets are known as replicas and are placed on various machines in the distributed file system. A replica contains the same information as the read-write fileset. The read-only fileset has a *.readonly* extension and it cannot be modified by normal file systems commands like `mkdir` and `rm`.

Replication is available in a cell if the following conditions are true:

- The cell's main read-write root.dfs fileset is a DCE LFS fileset. Note that replication is not available for non-LFS filesets.
- The cell's main read-write root.dfs fileset was mounted with an explicit read-write mount point.
- The root.dfs fileset is replicated.

In order to use read-only versions of a fileset, we must create read-only versions of all filesets mounted above it at higher levels in the file system. This means that we must create read-only copies of the filesets that contain its parent directories.

When the Cache Manager resolves a path name to locate a file, it begins at the top-level fileset in our cell and accesses the read-only versions of fileset whenever possible. If any fileset in the path name has no read-only version, the Cache Manager accesses only the read-write fileset versions until the file is resolved. Please see 3.3.3.3, "Types of Mount Points" on page 56 for a discussion of this algorithm.

Figure 37 on page 69 shows the DFS namespace when replication is being used. From this figure, we can see that:

- /:/usr and /:/src are regular mount points, /:/.usr is an explicit read-write mount point.
- root.dfs, root.dfs.readonly, usr, usr.readonly and src are filesets.
- Read-write filesets will be used to access and write files in the /:/.usr and /:/src directories.
- Read-only filesets will be used to access files in the /: directory, root.dfs.readonly fileset and in the /:/usr directory in the usr.readonly fileset.

The cache manager according to the algorithm presented in Figure 31 on page 57 will try to use files in the readOnly path whenever possible. Note that A is a read-write mount point while B and C are regular mount points. This allows the cache manager to use the readOnly versions whenever possible.

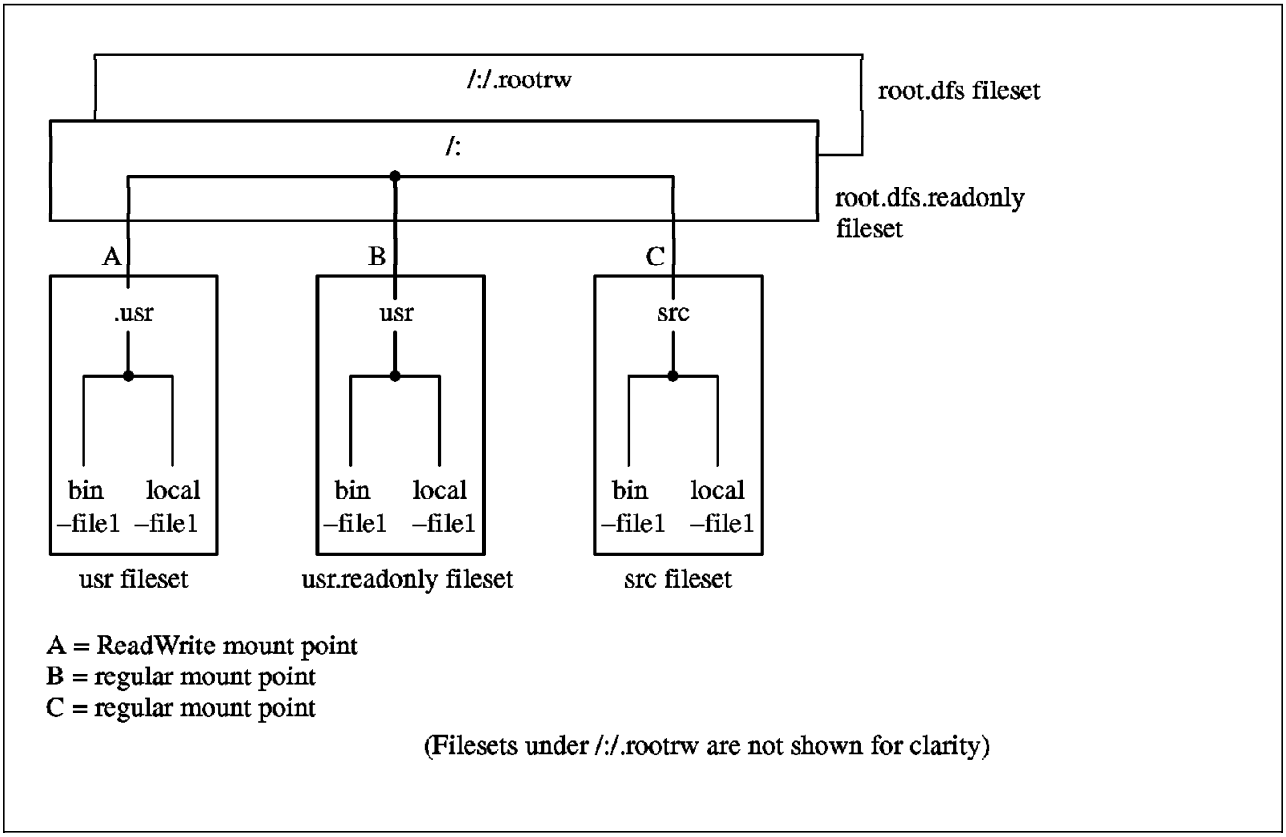


Figure 37. File System Hierarchy with Replicated Filesets

Note that in Figure 37, that:

- /:/.rootrw/usr/bin/file1 will access a read-write file
- /:/.usr/bin/file1 will access a read-write file
- /:/usr/bin/file1 will access a read-only file
- /:/src/bin/file1 will access a read-write file

The above figure can also be shown as in Figure 38 on page 70. This represents a logical view of the filespace. In this example, the administrator has explicitly mounted the read-write root.dfs fileset at /:/.rootrw and the usr read-write fileset at /:/.usr. The administrator then replicated the root.dfs and usr filesets. This represents the two different methods that administrators can use to mount the read-write filesets. In the method of replicating the /: (root) fileset, there is only one read-write access path to the replicated fileset data. Note that the /:/.usr/bin/file1 and the /:/.rootrw/usr/bin/file1 pathnames will both access the same file in the read-write fileset.

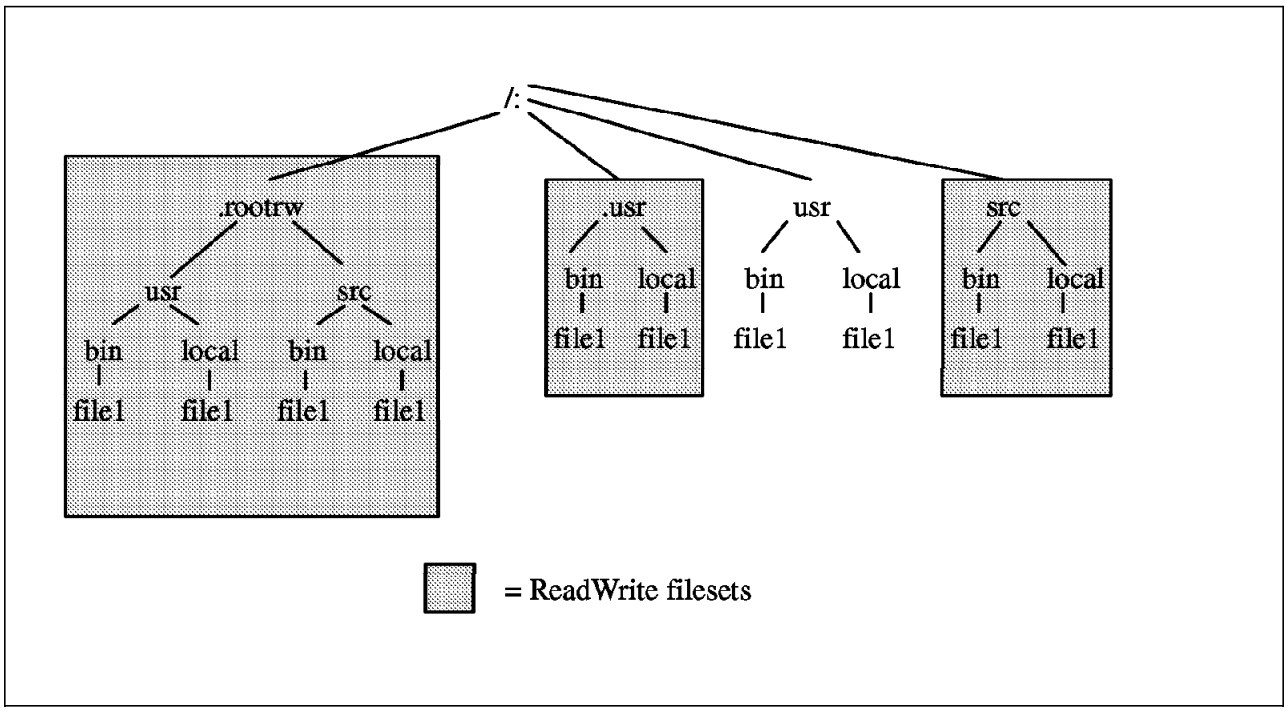


Figure 38. File System Hierarchy with Replicated Filesets

3.6.1 Creating Read-only DCE LFS Filesets.

Filesets that you may consider for replication should:

- Have files which are read or executed more frequently than modified
- Have files which are heavily used for example, executable files such as text editors, graphics editors or other popular programs
- Have files which need to be highly available so that if one machine goes down you can access the application program on another one

The Replication process can be started manually with Release Replication or automatically by using Scheduled Replication. In either case, you can manually cause an immediate update regardless of the type of replication you are using. This is done with the `fts update` command.

3.6.2 Release Replication (Manual Replication of Filesets)

Use Release Replication when you want to have control over when the data in the readOnly filesets match that of the readWrite filesets. You can also use release replication when contents of the filesets seldom change.

To use release replication you have to verify that you have the privilege needed such as inclusion in the admin.ft file on the machine where the read-write source is stored. Also verify your inclusion in the admin.fl files on all Fileset Database machines (or ownership of the machine where the read-write source is stored) and on each machine where a read-only replica is to reside.

The command `bos lsadmin -server <machine> -adminlist <filename>` will list the members of the various admin lists.

With Release Replication issue the `fts release` command to update read-only version of the read-write source fileset residing on the same File Server machine as the source fileset. The Replication Server *repserver* process at each replication site then places a copy of this read-only fileset at their respective sites. Release Replication is used to update the read-only copies of the read-write fileset upon demand.

The command `fts release -fileset {name | ID}` is used to initiate the release replication.

3.6.2.1 Configuring Filesets Using Release Replication

On the system which will have the read-write source fileset, do the following for release replication:

- Create and mount the readWrite fileset on the source machine.
- Create an explicit read-write mount point if you will need to update this fileset.
- Set the Replication parameters. (`fts setrepinfo` indicates the type of replication)
- Create a replica on the same file server as the source system. (`fts addsite` to add the replica on this system)

On each target site that will have a readOnly replica do the following:

- Create an aggregate large enough to house the fileset.
- Create the replica fileset. (`fts addsite` to add the readOnly replica on the target system)
- Send the replica to the target machines. (`fts release`)

3.6.3 Scheduled Replication (Automatic Replication of Filesets)

Use Scheduled Replication when you want to have the system automatically release the updates at a regular intervals and you do not care when the distribution of new replicas is done.

You can optionally specify additional parameters when you want to define Scheduled Replication. The Replication Server will then update the copies at your specified time interval. The `fts setrepinfo` command is used to set the replication parameters. The use of the defaults are recommended for Scheduled Replication.

3.6.3.1 Configuring Filesets Using Scheduled Replication

On the system which has the read-write fileset do the following sequence:

- Create and mount the read-write fileset on the source machine
- Set the replication parameters (`fts setrepinfo` to indicate scheduled replication)

On each system which will house a replica you must perform the following:

- Create an aggregate that is large enough to house the fileset.
- Create the replica fileset. (`fts addsite`)

All parameters set for the replication are stored in the FLDB entry. The replication will automatically occur according to the intervals you have set.

3.6.4 Immediate Update

Regardless whether you use release or scheduled replication, an immediate update of the read-only copies of a source fileset can be requested at any time with the `fts update` command. This command can be used to update either all replicas or to only update a replica at a specified site.

The command to immediate update is: `fts update -fileset {name|ID}{-all|-server machine}`

You need the name of the fileset or ID to be replicated. Use all to replicate all replicas or specify the DCE pathname of a specific File Server machine whose replica is to be updated.

3.6.5 Prerequisites for Replication

Before we can replicate a read-write fileset the following prerequisites must be met:

- Each File Server machine which will have a replica must have a server entry in the FLDB.
- The `fts setrepinfo` command sets or changes the replication parameters and stores the replication information in the FLDB entry for the fileset. This command is also used to indicate the type of replication being defined (release or scheduled). Only one type of replication is valid for a fileset at any particular time.
- A Replication Server process repserver and a Fileset Server process ftserver must be running on each File Server machine. The `bos status -server <machine>` command shows the processes that are running at the specified replication site.
- Use the `fts addsite` command to add the replication sites to the FLDB entry for the fileset. Use the `fts aggrinfo -server <machine >` command to check the available space on a target File Server machine's aggregate before defining replication sites. Use the `fts lsft` command to check the size of the read-write fileset.
- A read-only replica must also be stored on the same File Server machine as the read-write fileset with Release Replication.

3.6.6 Replication Type and Parameters

The syntax for the `fts setrepinfo` command is shown below:

```
fts setrepinfo -fileset  {name|ID} {-release|-scheduled} [-change] [-maxage
interval] [-failage interval] [reclaimwait interval] [-minrepdelay interval]
[-defaultSiteAge interval]
```

The following parameters are used for the `fts setrepinfo` command. They are described below:

- Option `-release`. It indicates release replication type.
- Option `-scheduled`. It indicates scheduled replication type.

These parameters are used both for Release or Scheduled replication:

- Parameter MaxAge. It specifies the maximum amount of time the Cache Manager distributes data without verifying if this data is still the same as the read-write version of the fileset.
- Parameter FailAge. It determines how long the Cache Manager continues to provide data obtained from a read-only replica. The Cache Manager will not give the data to an application if the data is older than the value and the Cache Manager cannot obtain more current data from the file server.
- Parameter ReclaimWait. It defines the amount of time the File Exporter waits before it reclaims storage space from deleted files. It also defines the frequency of keep-alive messages from Cache Manager to the Replication Server. These messages are used to indicate that the Cache Manager is still using files from a read-only replica.

These parameters are used only by Scheduled replication:

- Parameter MinRepDelay. It is used to specify how long the Replication Server waits after a read-write fileset changes before it attempts to get a new copy of the fileset.
- Parameter MaxSiteAge. It is used to control the maximum amount of time a replica can be out of date. This value must be stored with each site using the command fts addsite.
- Parameter DefaultSiteAge. The replication process will use the default value as the MaxSiteAge if that value is not set with the fts addsite command.

Table 5 will show the default parameters that are used for both scheduled and release replication.

Table 5 (Page 1 of 2). Descriptions of Replication Parameters

Command	Parameter	Default	Release Replication	Scheduled Replication
fts setrepinfo	MaxAge	2 hours	Need only if FailAge specified	Need only if FailAge,MinRepDelay, or DefaultSiteAge specified
fts setrepinfo	FailAge	1 day or twice MaxAge, which is larger	Optional	Need only if MinRepDelay, or DefaultSiteAge specified
fts setrepinfo	ReclaimWait	18 hours	Need only if FailAge specified	Need only if FailAge,MinRepDelay, or DefaultSiteAge specified
fts setrepinfo	MinRepDelay	5 minutes or one-quarter fo DefaultSiteAge, which is smaller	Not applicable	Need only if FailAge or DefaultSiteAge specified
fts addsite	MaxSiteAge	DefaultSiteAge	Not applicable	Need only if DefaultSiteAge not specified

Table 5 (Page 2 of 2). Descriptions of Replication Parameters				
Command	**Parameter**	**Default**	**Release Replication**	**Scheduled Replication**
fts setrepinfo	DefaultSiteAge	One-quarter of MaxAge	Not applicable	Optional

3.6.7 Replication Commands

These are the commands used in the replication environment:

- `fts addsite`: Adds a replication site for a read-write DCE LFS fileset
- `fts lsreplicas`: Displays the status of DCE LFS fileset replicas
- `fts release`: Starts Release Replication placing read-only version of a read-write DCE LFS fileset at the local site
- `fts rmsite`: Removes a replication site and read-only DCE LFS fileset
- `fts statrepserver`: Displays the status of a Replication server process
- `fts setrepinfo`: Sets or changes the replication type and parameters for a read-write DCE LFS fileset
- `fts update`: Requests an immediate update of replicas of a read-write DCE LFS fileset

Chapter 4. Installing and Configuring DFS

This chapter shows the installation and configuration of a DFS domain, including server and client components for DFS services. This chapter is designed to be used in conjunction with *AIX DCE Administration Guide*, which contains complete information about AIX DCE installation and configuration procedures.

The following sections provide:

- A general overview of DFS installation
- Step-by-step procedures for performing installation tasks
- A general overview of DFS configuration
- Step-by-step procedures for performing configuration tasks

4.1 Installation Overview

You must install the DCE DFS product on a machine that has previously been configured as a DCE client. In this section, we will present the software that is necessary for this.

4.1.1 AIX/6000 DCE Product Family

At the time this book was written (May, 1994) the AIX DCE product family consisted of the following software packages:

- AIX DCE Threads/6000 Version 1.1
- AIX DCE Base Services/6000 DCE Version 1.2
- AIX DCE Cell Directory Server/6000 Version 1.2
- AIX DCE Security Server/6000 Version 1.2
- AIX DCE Enhanced Distributed File System/6000 Version 1.1

The Threads/6000 package allows programmers to use threads without having the DCE software installed on a machine. The Base Services/6000 package includes the minimum set of software that is required in order to run the DCE on a machine. (You also get the Threads/6000 package when you order the Base Services/6000 package.) The Base Services/6000 package includes the threads, the software that is needed to be a DCE client and the basic DFS services. The Cell Directory Server/6000 and the Security Server/6000 are packages that perform the directory and security server functions. We will discuss both the Basic Services/6000 and the Enhanced Distributed File System/6000 software packages in this book. The Enhanced Distributed File System/6000 package contains the software that is necessary in order for a machine to be set up as a DCE LFS file server.

Why you need the Enhanced Distributed File System/6000

You can run the DCE Distributed File System without using the Enhanced Distributed File System/6000 software package. In this case, you will be able to export only the AIX Journaled File Systems (JFS) but not the DCE LFS file systems. The result is that you will not be able to have advanced functions of DCE LFS such as replication of filesets or the ability to have access on DCE LFS files limited to the granularity of a DCE principal. (In other words, ACLs on file system objects.)

We recommend that you use the Enhanced Distributed File System/6000 because this will save you the trouble of reconfiguring your DFS filespace if you desire replication of filesets at a later point in time.

4.1.2 Product Description of DFS and Enhanced DFS

There are three installable objects available as part of *IBM AIX DCE Product Family Version 1.2* :

dcebase.dfs.obj	DCE Basic DFS Services (part of AIX DCE Base Services/6000)
dceedfs.obj	DCE Enhanced DFS Services (part of AIX Enhanced DFS Services)
dceedfs.En_US.msg	DCE Enhanced DFS Messages (part of AIX Enhanced DFS Services)

When you order the AIX DCE Base Services/6000 licensed program you get the dcebase.dfs.obj binary. This binary provides the following functions:

- The AIX kernel extensions necessary for running the DFS client and the DFS file server
- SMIT screens for basic DFS system management
- AIX style system management commands (`mkdfs`, `mkdfsjfs`, and others)
- DFS processes and commands (flserver, ftserver, bosserver, dfsbind, dfsd, fxd and others)

The services of the basic distributed file system can be extended with the addition of dceedfs.obj and dceedfs.En_US.msg, which include a log-based physical file system, the DCE Local File System (LFS).

- The AIX kernel extension for running the DCE Local File System (LFS)
- SMIT screens for DCE LFS system management
- AIX style system management commands for DCE LFS (`mkdfslfs`, `rmdfslfs`, and others)
- Enhanced Distributed File System processes and commands such as bakserver, `newaggr` and `growaggr` as well as options for commands such as `bak`, `butc`, `fms` and `scout`. DFS related processes such as flserver, ftserver, bosserver, dfsbind, dfsd, fxd and others are also included.

As DFS uses the services of DCE such as Security Service, Cell Directory Service, Distributed Time Service and Remote Procedure Call. The machine must be minimally configured as a DCE client before it is configured to run DFS.

For further information on how to configure a DCE client, please refer to the "*AIX DCE Administration Guide*."

4.1.3 Prerequisite Software

Get the Latest Updates

Call your IBM support center to get the latest updates for the DCE and DFS products. By doing this, you will be able to get PTFs (software fixes) that are more current than the ones that are mentioned in this publication.

When you order DCE/DFS, you will get fixes called PTFs. These fixes will include fixes to the base operating system. Periodically contact your support structure for the latest PTFs.

The following table demonstrates the DCE installable objects and their prerequisite software:

Table 6. Installable DCE Objects and their Prerequisite Software

Installable Option	Prerequisite Software	Prereq sw description
dcepthreads.obj	bos.obj p=U418276	AIX Base Operating System V3.2
dcepthreads.En_US.msg	dcepthreads.obj	DCE Threads
dcebase.base.obj	bosnet.tcpip.obj dcepthreads.obj	AIX TCP/IP option DCE Threads
dcebase.En_US.msg	dcebase.base.obj	AIX DCE Base/6000
dcebase.admin.obj	dcebase.base.obj	AIX DCE Base/6000
dcebase.appdev.obj	dcebase.base.obj	AIX DCE Base/6000
dcebase.dfs.obj	dcebase.base.obj	AIX DCE Base/6000
dceedfs.obj	dcebase.base.obj	AIX DCE Base/6000
dceedfs.En_US.msg	dceedfs.obj	DCE Enhanced DFS Services

Table 7. Installable DCE Servers and their Prerequisite Software

Installable Option	Prerequisite Software	Prereq sw description
dcecds.En_US.msg	dcecds.obj	AIX DCE Cell Directory Servicer/6000
dcecds.obj	dcebase.base.obj	AIX DCE Base/6000
dcesec.En_US.msg	dcesec.obj	DCE Security Server
dcesec.obj	dcebase.base.obj	AIX DCE Base/6000

4.1.4 Required Disk Space (in Megabytes (MB))

DCE and DFS installable options require the following approximate amounts of disk space:

Table 8. Machine Type and Required Disk Space				
Installable Option	**DFS Server**	**DFS Client**	**DCE Security Server**	**CDS Server**
dcepthreads.obj	2.0	2.0	2.0	2.0
dcepthreads.En_US.msg	0.004	0.004	0.004	0.004
dcebase.base.obj	7.996	7.996	7.996	7.996
dcebase.admin.obj	2.2	2.2	2.2	2.2
dcebase.appdev.obj (3)	4.7	4.7	4.7	4.7
dcebase.dfs.obj	4.9	4.9	4.9	4.9
dcepriv.obj	0.2	0.2	0.2	0.2
dcebase.En_US.msg	0.22	0.22	0.22	0.22
dceedfs.obj	3.34			
dceedfs.En_US.msg	0.044			
dcesec.obj			2.6	
dcesec.En_US.msg			0.004	
dcecds.obj				1.4
dcecds.En_US.msg				0.004
Installed Subtotal	25.604	22.22	24.824	23.624
Configured	1.0	0.5	1.0	1.0
Total	26.604 (1)	22.72 (2)	25.824	24.624

Note: (1) These numbers do not include the disk space being exported by the DFS file server. This is variable depending upon your requirements.

Note: (2) In addition, DFS clients that keep their cache on disk require enough disk space to hold the cache. The size and location of the cache are selectable at configuration time. The recommended (and default) size of this DFS client cache is 10MB. In-memory caches require no additional disk space.

Note: (3) For installing DFS, the dcebase.appdev.obj binary is not necessary unless you intend to do DCE application development.

4.1.5 Paging Space

For a DFS client 32MB is the minimum amount of paging space that is recommended.

For a DFS server 100MB is the minimum amount of paging space that is recommended.

4.1.6 Real Memory Required

Base DFS clients require 16MB of memory, although 32MB is recommended. DFS servers require at least 32MB of memory.

4.2 DFS Installation - Step by Step

The system on which you will install DFS must have the proper prerequisite software installed. This section assumes this requirement has been satisfied prior to installing DFS. We will assume that your system is configured as a DCE client at a minimum.

The examples given throughout this chapter were done on a single machine. This machine was configured as a DCE cell. (It included the security server, the cell directory server and the distributed time server and could be used as a fully functional DCE client.

You can perform installation tasks using System Management Interface Tool (SMIT). SMIT uses interactive menus (rather than a command-line interface) to guide users through installation and other system management tasks. This section provides step-by-step procedures for DFS installation using SMIT.

Follow these steps on the machine that you want to run DFS on:

1. Login the system as root
2. Start SMIT:

   ```
   smit installp
   ```

3. Choose the “Install From All Available Software Packages” menu.
4. At the “INPUT device / directory for software” prompt, enter the name of device or directory which contains DFS installable objects. You can also press the **List** button to see which devices or directories are available and select one of them you want.
5. For the following screen:

   At the “SOFTWARE to install” prompt press the **List** button. If you want to run as a DFS client (or as a DFS server that will only be able to export AIX JFS file systems) select **dcebase.dfs.obj**. If you want to run as a DFS server that can export both JFS and DCE LFS filesets, select **dcebase.dfs.obj**, **dceedfs.obj** and **dceedfs.En_US.msg**.

```
                    Install From All Available Software Packages

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INPUT device / directory for software               /dev/rmt0
* SOFTWARE to install                                 []                      +
  Automatically install PREREQUISITE software?       yes                       +
  COMMIT software?                                   no                        +
  SAVE replaced files?                               yes                       +
  VERIFY Software?                                   no                        +
  EXTEND file systems if space needed?               yes                       +
  REMOVE input file after installation?              no                        +
  OVERWRITE existing version?                        no                        +
  ALTERNATE save directory                           []




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Undo             F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The system will have the proper DFS software installed on it. We can verify this as follows:

```
lslpp -h dce*

  dcebase.En_US.msg
             01.02.0000.0000 COMPLETE   APPLY      09/01/93   19:29:25 root
  dcebase.admin.obj
             01.02.0000.0000 COMPLETE   APPLY      09/01/93   19:29:25 root
  dcebase.base.obj
             01.02.0000.0000 COMPLETE   APPLY      09/01/93   19:29:24 root
     U419616 01.02.0000.0000 COMPLETE   APPLY      09/01/93   19:35:12 root
  dcebase.dfs.obj
             01.02.0000.0000 COMPLETE   APPLY      09/01/93   19:29:25 root
  dceedfs.En_US.msg
             01.01.0000.0000 COMPLETE   APPLY      09/01/93   19:35:39 root
  dceedfs.obj
             01.01.0000.0000 COMPLETE   APPLY      09/01/93   19:35:39 root
  dcepthreads.En_US.msg
             01.01.0000.0000 COMPLETE   APPLY      09/01/93   19:28:38 root
  dcepthreads.obj
             01.01.0000.0000 COMPLETE   APPLY      09/01/93   19:28:38 root
```

If you have other DCE servers such as a security server or a CDS server this will also be shown in the above list. We have edited these out for clarity.

This software must now be configured to perform the specific DFS operations.

4.3 Overview of DFS Configuration

Configuring DFS in a DCE cell requires the following components:

- System Control machine
- Fileset Database machine
- File Server machine

- DFS Client machine

A Backup Database machine could optionally be added to a DFS administrative domain.

DFS server and client machines run several different processes depending on how these machines are configured. For example, the *Basic OverSeer (BOS) Server*, or *bosserver* process 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. For more information on *Basic OverSeer (BOS) Server*, refer to 5.4.2, “BOS Server program” on page 136

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 an administrative list at any time. For more information on DFS security, refer to Chapter 6, “Security” on page 161.

The system administrator determines, at configuration, which processes are to be run on which machines. A machine′s role is determined by the types of processes it runs.

Before configuring a machine as a DFS server or client machines, you must have a DCE cell which addresses the following requirements:

- A Security Server must be running and available for the cell
- A CDS Server must be running and available for the cell
- At least three DTS (Distributed Time Service) Servers should be running and available for the cell
- Each DFS server or client machine must be configured at least as a DCE client. (You should be able to do a dce_login from that machine.) Please see ″Configuring and Starting Up DCE″ in the *AIX DCE Administration Guide*.

The following sections assume these requirements have been satisfied prior to configuring a machine as a DFS server or DFS client.

The following sections provide step-by-step procedures for configuring and administering a DFS domain by using the SMIT interface or the command line.

4.4 DFS Configuration for Servers and Clients - Step by Step

Configuring DFS in a DCE cell consists of the following steps. The steps should be followed in the order presented, although you can configure multiple machines of one role before moving on to the next. You can also come back later to configure another machine for a given role. For example, you can configure multiple Fileset Database machines before configuring the first File Server machine, but you can not configure a File Server machine before you have configured the first Fileset Database machine.

Configure the Machine as a DCE Client First

Make sure that you have DFS Server and DFS Client configured already as DCE clients. Each DFS server or client machine must 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 path name of each server machine, and a DFS server principal must also be created in the Registry Database for each server machine. (This should happen automatically.)

Please see the ″Configuring DCE Clients″ chapter of the *DCE Administration Guide* for a step-by-step procedure on how to set a DCE cell. This includes setting a machine up as a DCE client.

Step 1: Configure One or More System Control machines in the cell.

You can configure multiple domains by configuring multiple System Control machines. The domain that each DFS server machine belongs to is determined by the System Control machine that it receives its administration lists from. We recommend only one SCM per cell when first setting up your cell.

Step 2: Configure One or More Fileset Location Database Servers in the cell.

One Fileset Database machine is required in each DCE cell using DFS. Multiple Fileset Database machines in the cell provide availability and load balancing of the data. For most cells, three Fileset Database machines are recommended and sufficient. If more than three are configured, an odd number is preferred.

Step 3: Configure One or More File Server Machines

The File Server machines are configured by first starting the File Server daemons, then exporting DCE LFS aggregates and JFS file systems and their filesets. The root.dfs fileset must be created first. root.dfs is the root of the cell′s filespace and must be of an LFS type if replication is desired.

Step 4: Exporting Data from a DFS File Server

Prior to exporting a DCE LFS aggregate, you must use the `newaggr` command to construct the aggregate from a logical volume. The logical volume to be initialized as a DCE LFS aggregate must not be mounted locally or exported to the DCE namespace when you issue the `newaggr` command. Conversely, before exporting a non-LFS partition for use in DFS, you must create the partition and mount it locally.

Step 5: Configure One or More Backup Systems (Optional)

Backup Database machines are optional, and it is required only if you want to take advantage of DFS′s backup capabilities.

Configuring DFS Backup System includes the following tasks:

- Configure Backup Database machine
- Configure Tape Coordinator machine
- Define fileset families and fileset family entries
- Define a dump hierarchy of dump levels
- Label tapes (if necessary)

Step 6: Configure One or More DFS Clients in the Cell

A machine must be set up as a DFS client if you want to access the DFS namespace. The DFS client should be set up on any DFS machine where you need to access the DFS namespace. This includes the SCM, FLDB and file server machines.

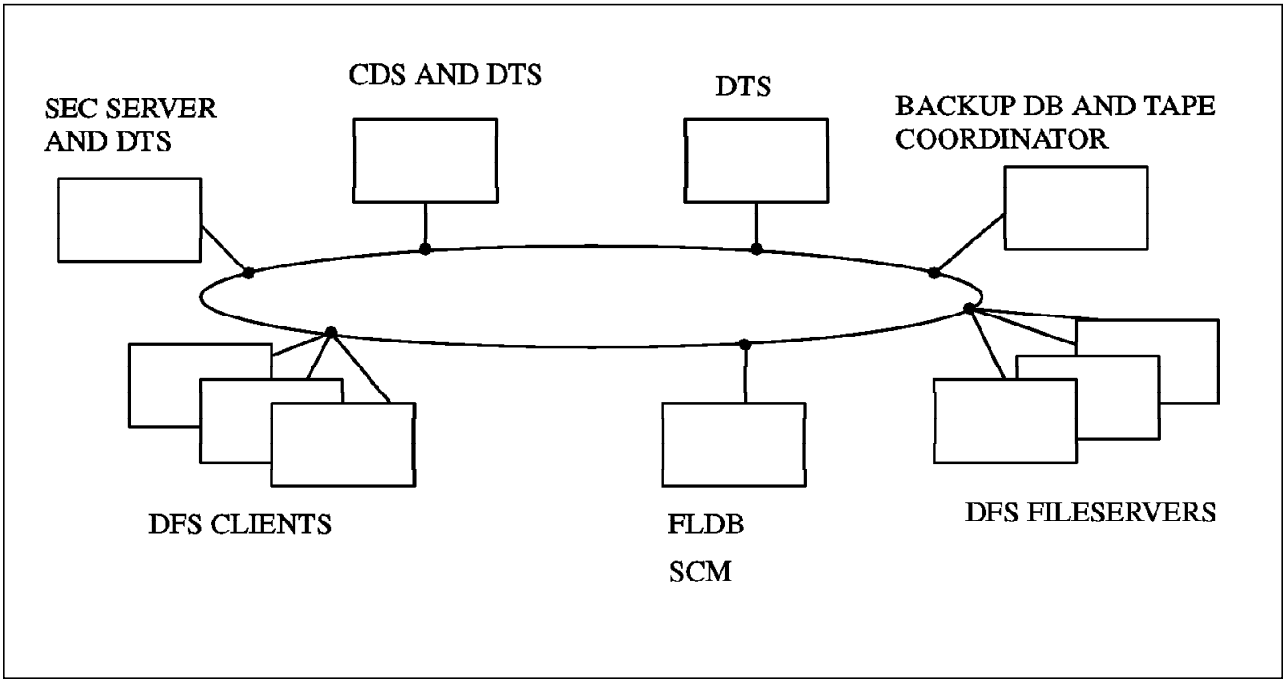


Figure 39. DCE Cell Configuration

The above picture shows the basic functions that are necessary in a DCE cell that has the DFS configured. First, there are requirements of a security server, a cell directory server and three timeservers. These requirements are all DCE requirements. Next, there are DFS file servers, DFS clients, at least one system control machine and at least one fileset location database server (FLDB). We will now show the steps to configure these DFS-related machines.

4.4.1 Step 1: Configuring the First System Control Machine in the Cell

To configure a machine as a DFS System Control machine, perform the following steps:

1. As root, start SMIT with `mkdfsscm`

   ```
   smit mkdfsscm
   ```

   or choose the following sequence of SMIT menu selections:

   a. *Communication Applications and Services*

   b. *DCE (Distributed Computing Environment)*

   c. *Configure DCE on the Local Machine*

   d. *Configure DCE Servers*

   e. *DFS System Control Machine*

2. If this machine has already been configured as a DCE client, the fields are already filled in as shown in the following screen:

```
                          DFS System Control Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* CELL name                                          [/.../dino]
* SECURITY server                                    [sys3]
  CDS server (If in a separate network)              []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.../dino/lan-profile]

F1=Help          F2=Refresh       F3=Cancel        F4=List
F5=Undo          F6=Command       F7=Edit          F8=Image
F9=Shell         F10=Exit         Enter=Do
```

If they are not, use the instructions in "Configuring DCE Clients" chapter of *DCE Administration Guide* to determine the values to enter.

3. Press **Enter**
4. When prompted, enter the cell administrator's password.

After configuring the SCM, you should see:

```
DFS System Control Machine (dfs_scm) configured successfully

Current state of DFS configuration:
dfs_scm      COMPLETE   DFS System Control Machine
          Press Enter to continue
```

If you cat /etc/mkdce.data:

```
# cat /etc/mkdce.data
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dfs_scm      COMPLETE   DFS System Control Machine
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Master Server
```

NOTE

You can also configure a machine as a DFS System Control machine through AIX command:

```
# mkdfs dfs_scm
```

Please see InfoExplorer for more information on parameters needed for this command.

4.4.2 Step 2: Configuring a DFS Fileset Database Machine

To configure a machine as a DFS Fileset Database machine, perform the following steps:

1. As root, start SMIT with `mkdfsfldb`

   ```
   smit mkdfsfldb
   ```

   or select the following sequence of SMIT menu options:

   a. *Communication Applications and Services*

   b. *DCE(Distributed Computing Environment)*

   c. *Configure DCE on the Local Machine*

   d. *Configure DCE Servers*

   e. *DFS Fileset Database Machine*

2. For the following screen:

   - If you want this machine to use DFS′s upclient process to receive its administration lists from a System Control machine, enter the name of that machine in the “DFS System CONTROL machine to get administration lists from” field. Use its DCE name (for example, /.:/hosts/sys1). If you leave this field blank, this machine maintains its own administration lists. If this machine has already been configured as a System Control machine, this field is ignored.
   - If you specified a System Control machine in the previous step and want to alter the amount of time the upclient process waits between checks for updated administration lists, change the value of the “FREQUENCY to update administration lists (in seconds)” field. The default is 300 seconds.
   - If you specified a System Control machine in the previous step and want this machine′s upclient process to log errors that occur while it requests administration lists from the System Control machine, enter the name of log file in the “LOG file for administration list updates” field. The directory where this file will be written must already exist.
   - If this machine has already been configured as a DCE client, the other fields are already filled in as shown in the following screen.

     If they are not, use the instructions in “Configuring DCE Clients” chapter of *DCE Administration Guide* to determine the values to enter.

```
                        DFS Fileset Database Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Additional GROUP to administer filesets on this ma []
  chine
  DFS System CONTROL machine to get                  []
      administration lists from
  FREQUENCY to update administration lists (in secon []
  ds)
  LOG file for administration list updates           []
* CELL name                                          [/.../dino]
* SECURITY server                                    [sys3]
  CDS server (If in a separate network)              []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.../dino/lan-profile]



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Undo             F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

3. Press **Enter**
4. When prompted, enter the cell administrator's password.

   It will take a few minutes to process. Please be patient.

You should see the following:

```
Configuring DFS Fileset Database Machine (dfs_fldb)...
Waiting (up to 5 minutes) for Fileset Database machines to elect a
synchronization site...
        /.../dino/hosts/sys4 has been elected synchronization site for the
Fileset Location Database.
DFS Fileset Database Machine (dfs_fldb) configured successfully

Current state of DFS configuration:
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
          Press Enter to continue
```

At this point, the following should be configured:

```
# cat /etc/mkdce.data
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
sec_srv      COMPLETE   Security Master Server
```

NOTE

You can also configure a machine as a DFS Fileset Database machine through AIX command:

```
# mkdfs -s /.:/hosts/sysX dfs_fldb
```

Please see InfoExplorer for more information on parameters needed for this command.

Replace sysX with the hostname of System Control machine in the DFS domain.

4.4.3 Step 3: Configuring a DFS File Server Machine

To configure a machine as a DFS File Server machine, perform the following steps:

1. As root, start SMIT with mkdfssrv

   ```
   smit mkdfssrv
   ```

   or select the following sequence of SMIT menu options:

   a. *Communication Applications and Services*

   b. *DCE(Distributed Computing Environment)*

   c. *Configure DCE on the Local Machine*

   d. *Configure DCE Servers*

   e. *DFS File Server Machine*

2. For the following screen:

   - If you want members of a certain DCE group to have administration authority over the filesets on this machine, specify that group′s name in the “Additional GROUP to administer filesets on this machine” field. Principals and groups specified in the admin.ft administration lists have authority on all filesets on all File Server machines in the domain; this field allows you to specify an additional group that only has authority on this machine.
   - In the event that a DCE cell has to be reconfigured, it is possible to recover and reuse the DFS aggregates and filesets from the original cell. When the DFS File Server machine is reconfigured in the new cell, the aggregates listed in the /var/dce/dfs/dfstab file will be exported by DFS. Remember that the DCE LFS kernel extension should be loaded before exporting the aggregates. You can load DCE LFS kernel extention by filling “ yes” in the “Load DCE LFS kernel extension?” field at this time.
   - If you want this machine to use DFS′s upclient process to receive its administration lists from a System Control machine, enter the name of that machine in the “DFS System CONTROL machine to get administration lists from” field. Use its DCE name (for example, /.:/hosts/sys1). If you leave this field blank, this machine maintains its own administration lists. If this machine has already been configured as a System Control machine, this field is ignored.

- If you specified a System Control machine in the previous step and want to alter the amount of time the upclient process waits between checks for updated administration lists, change the value of the “FREQUENCY to update administration lists (in seconds)” field. The default is 300 seconds.
- If you specified a System Control machine in the previous step and want this machine’s upclient process to log errors that occur while it requests administration lists from the System Control machine, enter the name of log file in the “LOG file for administration list updates” field. The directory where this file will be written must already exist.
- If this machine has already been configured as a DCE client, the other fields are already filled in like following screen.

  If they are not, use the instructions in “Configuring DCE Clients” chapter of *DCE Administration Guide* to determine the values to enter.

```
                          DFS File Server Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
  Additional GROUP to administer filesets on this ma []
  chine
  Load LFS kernel extension?                        [no]                    +
  DFS System CONTROL machine to get                 []
      administration lists from
  FREQUENCY to update administration lists (in secon []
  ds)
  LOG file for administration list updates          []
* CELL name                                         [/.../dino]
* SECURITY server                                   [sys3]
  CDS server (If in a separate network)             []
* Cell ADMINISTRATOR's account                      [cell_admin]
* LAN PROFILE                                       [/.../dino/lan-profile]



F1=Help           F2=Refresh         F3=Cancel          F4=List
F5=Undo           F6=Command         F7=Edit            F8=Image
F9=Shell          F10=Exit           Enter=Do
```

3. Press **Enter**
4. When prompted, enter the cell administrator’s password.

   It will take a few minutes to process. Please be patient.

This procedure only configures the processes necessary for the File Server, but no data is yet being exported from the machine to clients. The following section details the information on how to export data. (At this time you may want to make sure that the /opt/dcelocal/var/dfs/dfstab file has been deleted if you are doing a reconfiguration and you do not want to have the aggregate information that had been contained in the dfstab file.)

If a dfstab file exists you will see the following result:

```
File /opt/dcelocal/var/dfs/dfstab already exists--check it for old entries
after configuration is complete.
 Password must be changed!
 Configuring DFS File Server Machine (dfs_srv)...
 DFS File Server Machine (dfs_srv) configured successfully
```

```
Current state of DFS configuration:
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
dfs_srv      COMPLETE   DFS File Server Machine
          Press Enter to continue

# cat /etc/mkdce.data
cds_cl       COMPLETE   CDS Clerk
cds_srv      COMPLETE   Initial CDS Server
dfs_fldb     COMPLETE   DFS Fileset Database Machine
dfs_scm      COMPLETE   DFS System Control Machine
dfs_srv      COMPLETE   DFS File Server Machine
dts_local    COMPLETE   Local DTS Server
rpc          COMPLETE   RPC Endpoint Mapper
sec_cl       COMPLETE   Security Client
```

Note

You can also configure a machine as a DFS File Server machine through AIX command:

```
# mkdfs -s /.:/hosts/sysX dfs_srv
```

Replace sysX with the hostname of System Control machine in the DFS domain.

Please see InfoExplorer for more information on parameters needed for this command.

4.4.4 Step 4: Exporting Data from a DFS File Server

Once you have configured a machine as a DFS File Server, you have to export either a DCE LFS aggregate or a JFS partition from a File Server machine for use as the root fileset of the cell. This section describes the steps involved in initializing and exporting both a DCE LFS aggregate and a JFS partition.

The first step you have to do for setting up the DFS namespace from File Server machine is configuring root.dfs fileset which is the top level fileset in the distributed file system. root.dfs is analogous to the “/”(root) file system device on a traditional UNIX system and so used for mounting other filesets.

Note

You can configure either an DCE LFS fileset or a non-LFS fileset as a root.dfs, but it is strongly recommended that root.dfs be an DCE LFS fileset. Only an DCE LFS fileset can support the replication feature of DFS. If the root file system is *not* a DCE LFS fileset, then any subsequent filesets cannot be replicated.

You can create the root.dfs fileset either using the command line or from a SMIT interface.

4.4.4.1 Configuring a DCE LFS root.dfs from SMIT

You can also configure the DCE LFS root.dfs fileset from SMIT as follows:

1. Create a logical volume that will contain the DCE LFS root.dfs fileset for the cell.

   ```
   # smit mklv
   ```

   For the following screen:

   - It is useful to set the logical volume name to lfsrootlv to make it easier to identify it as a **root.dfs** device.
   - It is recommended to set the logical volume type to LFS to make it easier to identify it as a DCE LFS aggregate.

```
                              Add a Logical Volume

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
  Logical volume NAME                              [lfsrootlv]
* VOLUME GROUP name                                 rootvg
* Number of LOGICAL PARTITIONS                     [1]   #
  PHYSICAL VOLUME names                            []                      +
  Logical volume TYPE                              [lfs]
  POSITION on physical volume                       midway                 +
  RANGE of physical volumes                         minimum                +
  MAXIMUM NUMBER of PHYSICAL VOLUMES               []                       #
    to use for allocation
  Number of COPIES of each logical                  1                      +
    partition
  Mirror Write Consistency?                         yes                    +
  Allocate each logical partition copy              yes                    +
    on a SEPARATE physical volume?
  RELOCATE the logical volume during                yes                    +
    reorganization?
  Logical volume LABEL                             []
  MAXIMUM NUMBER of LOGICAL PARTITIONS             [128]
  Enable BAD BLOCK relocation?                      yes                    +
  SCHEDULING POLICY for writing logical             parallel               +
    partition copies
  Enable WRITE VERIFY?                              no                     +
  File containing ALLOCATION MAP                   []




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Undo             F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

   a.

   ```
   # dce_login cell_admin -dce-
   ```

      - Authenticates as a DCE user with proper authority to perform DFS aggregate and fileset tasks.

   b.

   ```
   # smit dfslfs
   ```

   or choose the following sequence of SMIT menu selections:

      a. *Communication Applications and Services*

      b. *DCE(Distributed Computing Environment)*

c. *DFS(Distributed File System) Administration*

d. *Add/Delete DCE LFS Aggregates and Filesets*

c. Choose the **Export an Aggregate from the Local Machine** menu option.

d. At the following screen:

- Select **no** if you are about to export a logical volume that has already had the newaggr command run on it.
- Select **yes** if you are about to export a logical volume that has just been created and has never had the newaggr command run on it, or you want to remove the data stored on it by reinitializing it. In our case, we select yes because we have just created the logical volume.

```
                        INITIALIZE device for LFS?

  Move cursor to desired item and press Enter.

    1 no
    2 yes

  F1=Help                 F2=Refresh              F3=Cancel
  F8=Image                F10=Exit                Enter=Do
```

e. At the following screen:

- In the “DEVICE to export as aggregate” field, give the device name for the logical volume (for example, /dev/lfsrootlv).
- In the “Aggregate NAME” field, give the name you want the aggregate to be called (for example, lfsroot).
- In the “Aggregate ID” field, specify the ID to be used for this aggregate. This number has to be unique among all aggregates on the machine. If you leave this field blank, the system will take the next number in sequence automatically.
- For the other prompts, take the default value.

```
                  Export an Aggregate from the Local Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* DEVICE to export as aggregate                     [/dev/lfsrootlv]
* Aggregate NAME                                    [lfsroot]
  Aggregate ID                                      [1]
  BLOCK size (in bytes)                             [8192]                +
  FRAGMENT size (in bytes)                          [1024]                +
  Normal or verbose OUTPUT during                   [normal]              +
    device initialization?

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Undo            F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

f. Press **Enter**

At this time, the DCE LFS aggregate is initialized and exported. No fileset have been created in the aggregate yet.

g. Press **F10**

h.

```
# smit mkdfsft
```

i. Select the **Create a Fileset in an Aggregate on the Local Machine** menu option.

j. At the following screen:

- In the “FILESET name” field, enter **root.dfs**
- In the “AGGREGATE to contain fileset” field, give the name of aggregate in which the root.dfs fileset should be created (for example, lfsroot).
- In the “MOUNT POINT for fileset” field, leave this field blank. The root.dfs fileset is automatically mounted at /.:/fs when it is defined.

```
            Create a Fileset in an Aggregate on the Local Machine


Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* FILESET name                                     [root.dfs]
* AGGREGATE to contain fileset                     [lfsroot]              +
  MOUNT POINT for fileset                          []









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Undo             F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

k. Press **Enter**

At this point, the DCE LFS root.dfs fileset is created as you specified.

4.4.4.2 Configuring an DCE LFS root.dfs Fileset from the Command Line

1. Create a logical volume that will contain the DCE LFS root.dfs fileset for the cell.

```
# mklv -y lfsrootlv -t lfs rootvg 1
lfsrootlv
```

2. Create a DCE LFS aggregate on this logical volume

```
# newaggr -aggregate /dev/lfsrootlv -blocksize 8192 -fragsize 1024 \
-overwrite
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 511
*** Defaulting to 13 log blocks    (maximum of 1
concurrent transactions).
```

```
/dev/rlfsrootlv: Marked as not a BSD file system any more.
Done.  /dev/rlfsrootlv is now an Episode aggregate.
```

The syntax for the newaggr command is as follows:

 # newaggr -aggregate *name* **-blocksize** *bytes* **-fragsize** *bytes* [**-overwrite**]

- -aggregate *name*: This is the device name of the AIX logical volume you just created in the previous step (for example, /dev/lfsrootlv).
- -blocksize *bytes*: This is the number of bytes in each block used for the DCE LFS. Allowable values are powers of two from 4096 to 65,536 but 8192 bytes is the recommended block size.
- -fragsize *bytes*: This is the number of bytes in a DCE LFS fragment. The valid range is from 1024 to the number specified with -blocksize.
- [-overwrite]: (optional) This specifies that an existing file system found on the partition can be overwritten. You must specify the -overwrite option to force the action if you get the following error message:

  ```
  *** /dev/rlfsrootlv CONTAINS A BSD FILE SYSTEM WITH DATA
  *** Using default initialempty value of 1.
  *** Using default number of (8192-byte) blocks: 511
  *** Defaulting to 13 log blocks    (maximum of 1 concurrent
  transactions).
  Problem with making /dev/rlfsrootlv an Episode aggregate.
  ```

In general, the defaults for the newaggr command will be sufficient.

3. Configure the root.dfs fileset. Note that you must be DCE logged in as cell_admin. (Configuring the root fileset includes exporting the aggregate, creating the DCE LFS fileset, and mounting the DCE LFS fileset.)

Note

The mkdfslfs command is an IBM command that simplifies the configuration of an DCE LFS root.dfs by taking care of several important steps in one command. The mkdfslfs command:

- Loads the DFS kernel extention if this is not already loaded.
- Creates the /opt/dcelocal/var/dfs/dfstab file and adds an entry for the aggregate.
- Exports the aggregate (# dfsexport).
- Creates the DCE LFS fileset (# fts create).

 # mkdfslfs -r -d *device_name* **-n** *aggregate_name*

- -r This specifies that the root fileset (root.dfs) should be created in the aggregate that you specify.
- -d *device_name*: It is the name of the block device for the DCE LFS aggregate being exported (for example, /dev/lfsrootlv). This device should already have had the newaggr command run on it.
- -n *aggregate_name*: It is the name of the aggregate (for example, lfsroot).
- You don't have to specify -m *DFS_mount_point* option at this time, because root.dfs fileset is automatically mounted at **/.:/fs** when it is defined.

Therefore, the following command creates a fileset named root.dfs on the lfsroot aggregate. This aggregate is located on the block device /dev/lfsrootlv and the root.dfs fileset will be mounted on the /.:/fs mount point.

```
# mkdfslfs -r -d /dev/lfsrootlv -n lfsroot
    readWrite   ID 0,,1  valid
    readOnly    ID 0,,2  invalid
    backup      ID 0,,3  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

sys4                RW       lfsroot 0:00:00 hosts/sys4     <nil>

Fileset 0,,1 created on aggregate lfsroot of /.:/hosts/sys4
```

At this time you should be able to do the following things:

- cat /var/dce/dfs/dfstab and see that the entry for the lfsrootlv exists.
- run dfsexport to see what aggregates and JFS partitions your file server has already exported.
- run fts lsfldb to see what fileset the FLDB actually knows about

Whenever you have a command like mkdfslfs that does several things, it is important to know what it is actually accomplishing. This is because sometimes things will go wrong and the command will only partially complete. Here are the basics on what this command does as well as some hints on what to do if things do not go right. (For example, you forget to dce_login before you use this command.)

1) It creates the /opt/dcelocal/var/dfs/dfstab file if this does not exist and creates the entry specified with the -d option. This file can also be accessed as /var/dce/dfs/dfstab. The purpose of this file is to list all of the DCE LFS aggregates and JFS partitions on the particular file server that CAN be exported into the DFS filespace.

2) It runs dfsexport to export the aggregate into the DFS filespace. This takes the DCE LFS aggregate or that JFS partition that you want to export and makes an entry in the file servers /var/dce/dfs/dfsatab file. The /var/dce/dfs/dfsatab file contains entries that the file server has already exported into the DFS namespace. These aggregates and partitions are then available to the DFS clients.

3) It runs fts create to create a fileset on the aggregate and server that you specify. In this case, when we use the -r option, it creates the root.dfs fileset. At this time, an entry is also made in the FLDB (fileset location database) so that DFS clients can find the location of the files in the fileset.

4) The root.dfs fileset is automatically mounted at /.:/fs

Removing Aggregates and Partitions from the DFS Namespace

In case you have to remove some aggregates and partitions from the DFS Namespace, here are some tips on how to do this:

To list the aggregates currently exported by a server:

```
fts lsaggr -server /.:/hosts/<servername>
```

To detach the aggregate from the DFS namespace:

```
dfsexport -aggregate <aggregatename> -detach
```

This will unexport the aggregate from the DFS namespace.

To remove the fileset from the aggregate and from the FLDB:

```
rmdfslfs -f <filesetname>
```

To also remove the mount point:

```
rmdfslfs -f <filesetname> -m <DFS_mount_point>
```

4.4.4.3 Successful Completion of the DCE LFS root.dfs Fileset

If you have successfully followed the above steps with success, you can see the following results when you issue the following:

```
# pg /var/dce/dfs/dfstab
# blkdev         aggname                 aggtype aggid   [UFS fsid]
/dev/lfsrootlv lfsroot                lfs     1

# dfsexport

dfsexport: /dev/lfsrootlv, lfs, 1, 0,,0
# fts lsfldb
root.dfs
        readWrite   ID 0,,1  valid
        readOnly    ID 0,,2  invalid
        backup      ID 0,,3  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner
sys1                RW,BK    lsf.root 0:00:00 hosts/sys1     <nil>


----------------------
Total FLDB entries that were successfully enumerated: 1 (0 failed; 0 wrong aggr
 type)
```

4.4.4.4 Configuring a non-LFS root.dfs

If you do not require the ACL, replication and other DCE LFS fileset capabilities you do not need to purchase the AIX DCE Enhanced Distributed File System/6000 Version 1.1 program product. You could set up a DFS environment by just using the AIX DCE Base Services/6000 DCE Version 1.2. Please see the discussion on page 76. This will let you know what limitations exist for using JFS as the root.dfs fileset. The steps below show how to set up a DFS using JFS as the root fileset.

1. Create the AIX JFS file system.

   ```
   # smit crjfs
   ```

   At the following screen:

   - Use the default rootvg as the volume group name.

- Set the “Mount AUTOMATICALLY at system restart?” field to ′yes′.

```
                        Add a Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
Volume group name                                    rootvg
* SIZE of file system (in 512-byte blocks)           [8000]                  #
* MOUNT POINT                                        [/export/jfsroot]
  Mount AUTOMATICALLY at system restart?             yes                    +
  PERMISSIONS                                        read/write             +
  Mount OPTIONS                                      []                     +
  Start Disk Accounting?                             no                     +

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Undo            F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

2. Find the name of the logical volume that the fileset was created on.

   You can use the `lsfs -a` command to list the names of the file systems. By searching for the mount point, you can find the entry that contains the name of the logical volume that was used. In our example, we have the following:

   ```
   # lsfs -a |grep /export/jfsroot
   /dev/lv03       --         /export/jfsroot        jfs   8192    rw         yes
   no
   ```

3. The JFS file system must now be mounted before you can add this to the DFS namespace.

   ```
   # mount /export/jfsroot
   ```

4. Log into DCE.

   ```
   dce_login cell_admin -dce-
   ```

5. Configure the **root.dfs** fileset

   ```
   # mkdfsjfs -r -d /dev/lv04
   ```

6. Verify that everything worked:

   ```
   # df
   Filesystem     Total KB     free %used   iused %iused Mounted on
   /dev/hd4           4096      920   77%     751    73% /
   /dev/hd9var       16384    13660   16%    1575    38% /var
   /dev/hd2         253952    52756   79%   12267    19% /usr
   /dev/hd3           8192     1500   81%      45     2% /tmp
   /dev/hd1          20480    19636    4%      44     0% /home
   /dev/lv03         28672     6276   78%      27     0% /usr/sys/inst.images
   /dev/lv04          4096     3936    3%      16     1% /export/jfsroot

   # cat /var/dce/dfs/dfstab
   # blkdev        aggname aggtype aggid   [UFS fsid]
   /dev/lv04       /export/jfsroot ufs     1       0,,13
   # dfsexport
   ```

```
dfsexport: /dev/lv04, ufs, 1, 0,,13
# fts lsfldb

root.dfs
        readWrite   ID 0,,13  valid
        readOnly    ID 0,,14  invalid
        backup      ID 0,,15  invalid
        number of sites: 1
        server      flags     aggr   siteAge principal      owner

sys4                RW       /export/jfsroot 0:00:00 hosts/sys4     <nil>

----------------------
Total FLDB entries that were successfully enumerated: 1 (0 failed; 0 wrong aggr
type)
```

Note

The mkdfsjfs command is an IBM command that simplifies the configuration of a JFS fileset by taking care of several steps in one command:

- Figures out a unique aggregate ID on your system for you and adds an entry into the dfstab file for the aggregate. You can do this yourself with the *-i* option.
- Creates an entry for the fileset in the FLDB (# fts crfldbentry)
- Exports the partition (# dfsexport).
- Creates the mount point for the fileset (# fts crmount).

The syntax for the mkdfsjfs command is as follows:

mkdfsjfs -r -d *device_name*

- -r This specifies that the root fileset (root.dfs) should be created in this partition.
- -d *device_name*: It is the name of the block device for the JFS partition being exported (for example, /dev/lv00). You can get *device_name* from looking at the /etc/filesystems file.
- You don't have to specify -m *DFS_mount_point* option at this time, because root.dfs fileset is automatically mounted at /.:/fs when it is defined.

Assuming that your machine has been defined as a dfs_client, you should now be able to cd to the /: directory. If this is not the case and the machine does not recognize the /: directory as being a valid directory, you may be in Token State Recovery mode (TSR). This mode could last on the order of minutes before the /: directory is recognized as being valid.

You can also do step 5 using SMIT:

1. # dce_login cell_admin -dce-
2. # smit dfsjfs

   or select the following sequence of SMIT menu options:

   a. *Communication Applications and Services*
   b. *DCE(Distributed Computing Environment)*
   c. *DFS(Distributed File System) Administration*
   d. *Add/Delete JFS File Systems*

3. Select the **Export a File System from the Local Machine** menu option.
4. At the following screen:
   - In the “DEVICE to export” field, give the name of the device on which the JFS file system is located (for example, /dev/lv04).
   - In the “FILESET to register in file system as” field, enter **root.dfs**.
   - In the “MOUNT POINT for fileset” field, leave this field blank.
   - In the “Aggregate ID to assign to file system” field, specify the ID to be used for this partition. If you leave this field blank, the next available ID number is used.

```
                  Export a File System from the Local Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* DEVICE to export                                     [/dev/lv04]
* FILESET to register in file system as                [root.dfs]
  MOUNT POINT for fileset                             []
  Aggregate ID to assign to file system               [1]

F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Undo           F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

5. Press **Enter**.

   At this point, the JFS file system is exported as root.dfs fileset.

4.4.4.5 Adding a DCE LFS Fileset into the DFS Filespace

At this point, the root.dfs fileset should be installed into the DFS namespace. You may want to verify this by using the `fts lsfldb` command to check for the lfsroot or jfsroot file system. Your machine must also be configured as a DFS client so that you can have access to the DFS namespace. In other words, you need to be configured as a DFS client so that you can cd /: and mount filesets into the DFS namespace. At this point, we will now show how you can mount LFS aggregates and filesets into the DFS namespace. As you recall, a DFS aggregate can have more than one fileset in it. (It can also have zero.) In Figure 40 on page 99 you can see an aggregate that has more than one fileset in it.

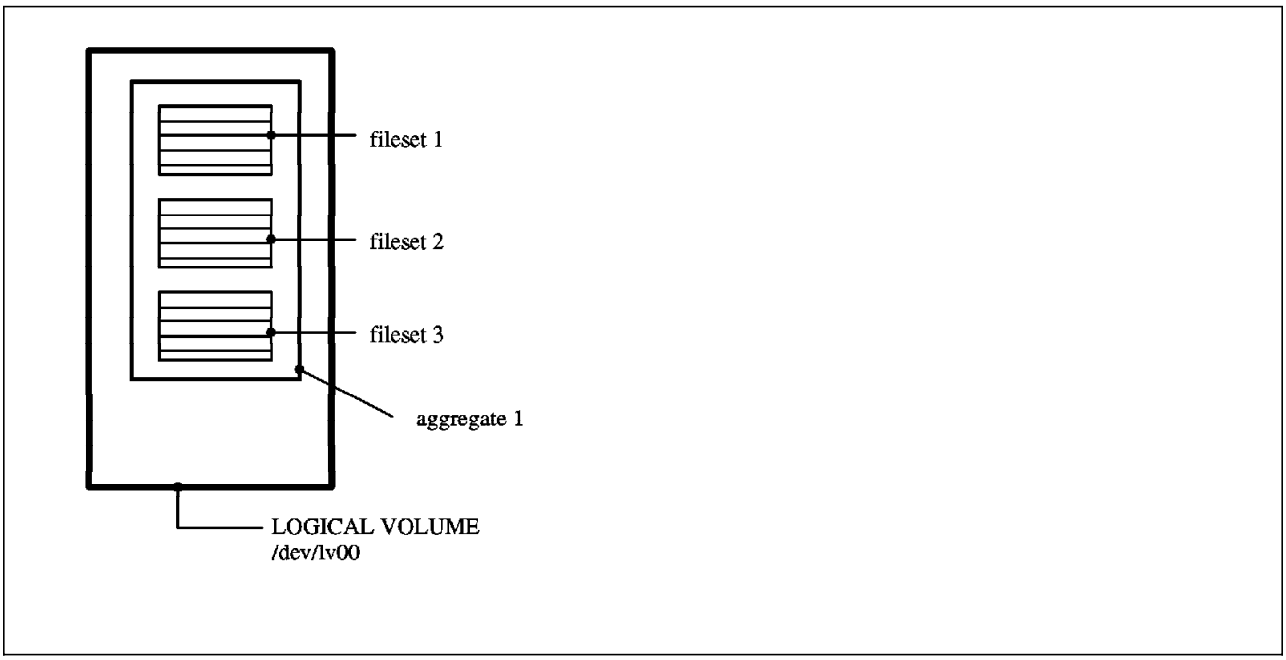


Figure 40. Multiple DCE LFS Filesets can be Stored in an Aggregate

Adding a DCE LFS Fileset

1. Login as native operating system root user on File Server machine. Also do a dce_login as a principal that is in the dfs-admin group (cell_admin has this capability)
2. Create the aggregate which is to contain a DCE LFS Fileset.
   - You can use an existing aggregate. An aggregate can house multiple DCE LFS Filesets.
   - You may create a new logical volume and initialize an aggregate on this logical volume (`newaggr` command).
3. Create and configure an DCE LFS Fileset on the aggregate which you prepared in the previous step.

   The `mkdfslfs` command is used to accomplish this. The parameters that you use in this command depend on whether you are exporting a new aggregate, creating a new fileset (as opposed to just exporting an empty aggregate) and whether you want the fileset to be immediately mounted. Please refer to Figure 41 on page 100 for a visual explanation of the command.

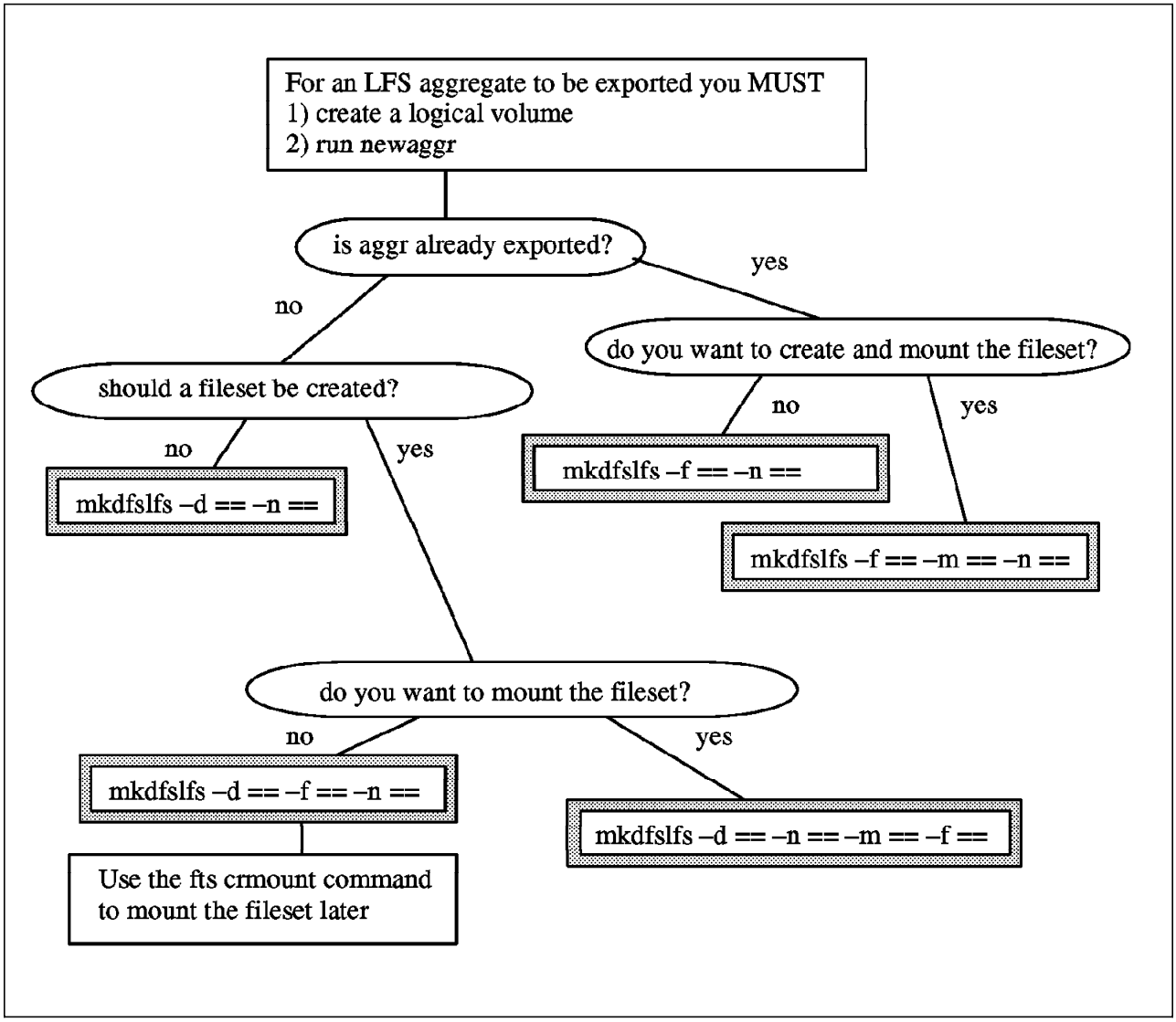


Figure 41. Using the mkdfslfs Command

The syntax for the `mkdfslfs` command is shown below.

`mkdfslfs -f` *fileset_name* `-m` *DFS_mount_point* `-d` *device_name* `-n` *aggregate_name*

- `-f` *fileset_name*: The name of the fileset to be defined in this partition. It should be unique with the cell.
- `-m` *DFS_mount_point*: Specifies the location in the DFS file space where the fileset should be mounted (for example, /.:/fs/dir1). The issuer of a `mkdfslfs` or `mkdfsjfs` command that uses the `-m` option must have the following permissions; If the parent directory of the mount point is in a DCE LFS fileset, the issuer must have *write*, *execute:ehp1, control and insert permissions on the directory. If the directory is in a non-LFS fileset, the issuer must have write and execute permissions on the directory.*
- `-d` *device_name: The name of the block device on which the aggregate to be exported resides.*
- `-n` *aggregate_name: The name of the aggregate that is to be exported.*

Here is an example of adding a new aggregate and fileset to the DFS namespace.

1) Create a logical volume for the new LFS fileset:

```
# mklv -y lfs1lv -t lfs rootvg 1
lfs1lv
```

2) Create a new aggr on that root fileset:

```
# newaggr -aggregate /dev/lfs1lv  \
 -blocksize 8192 -fragsize 1024 -overwrite
```

3) Use the mkdfslfs to export the aggregate and fileset and to do the mount.

```
# mkdfslfs -f lfs.one -m /:/dir1 -n aggr1 -d /dev/lfs1lv
        readWrite   ID 0,,19  valid
        readOnly    ID 0,,20  invalid
        backup      ID 0,,21  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

sys4                RW       aggr1   0:00:00 hosts/sys4     <nil>

Fileset 0,,19 created on aggregate aggr1 of /.:/hosts/sys4
```

(Note that the /:/dir1 directory must not be created before you issue this command.) From a machine that has been configured as a DFS client, you can now:

```
# cd /:/dir1
```

- *For more information on other options, see InfoExplorer.*

4.4.4.6 Adding a JFS Fileset into the DFS Filespace

Once again, a root.dfs fileset must already exist in the DFS namespace prior to exporting a JFS fileset into the DFS namespace. With JFS file systems, you can only have one per logical volume. Thus you do not have the concept of an aggregate and multiple filesets on that aggregate. Figure 42 shows this one file system per logical volume relationship.

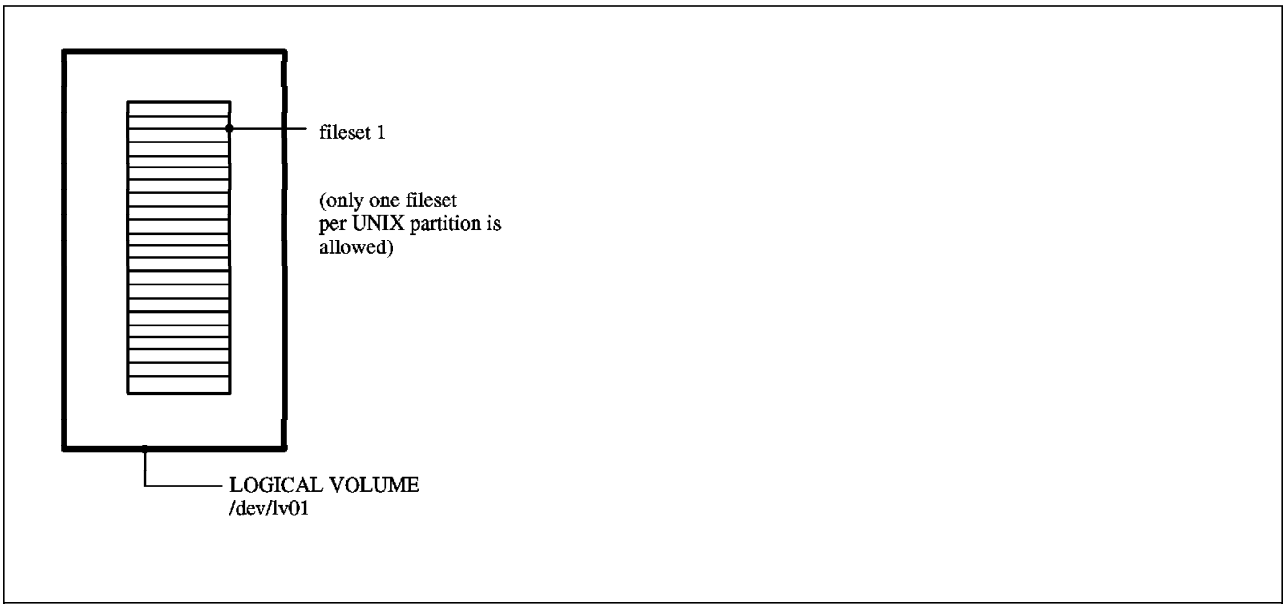


Figure 42. Only One jfs Fileset Can Be Stored on a Disk Partition.

Adding a JFS Fileset: Adding a JFS fileset or file system to the DCE Namespace is similar to the way the LFS filesets were added.

1. Login to dce as cell_admin (`# dce_login cell_admin -dce-`)
2. Create and mount a JFS file system.
   - You can also use existing JFS file systems for export to the DFS file space. There should not be any file system activity occurring on that file system. Local activity on AIX JFS file system causes `dfsexport` to fail. To create a new file system, use `smit crjfs`.
3. Export a JFS file system and define it as a fileset.

   Since the JFS file systems do not have the concepts of aggregates and multiple filesets per aggregate, it is much simpler. The `mkdfsjfs` command is used to define the file system as a fileset and to optionally mount it into the DFS namespace at this time. See Figure 43 for an overview to understanding the `mkdfsjfs` command.

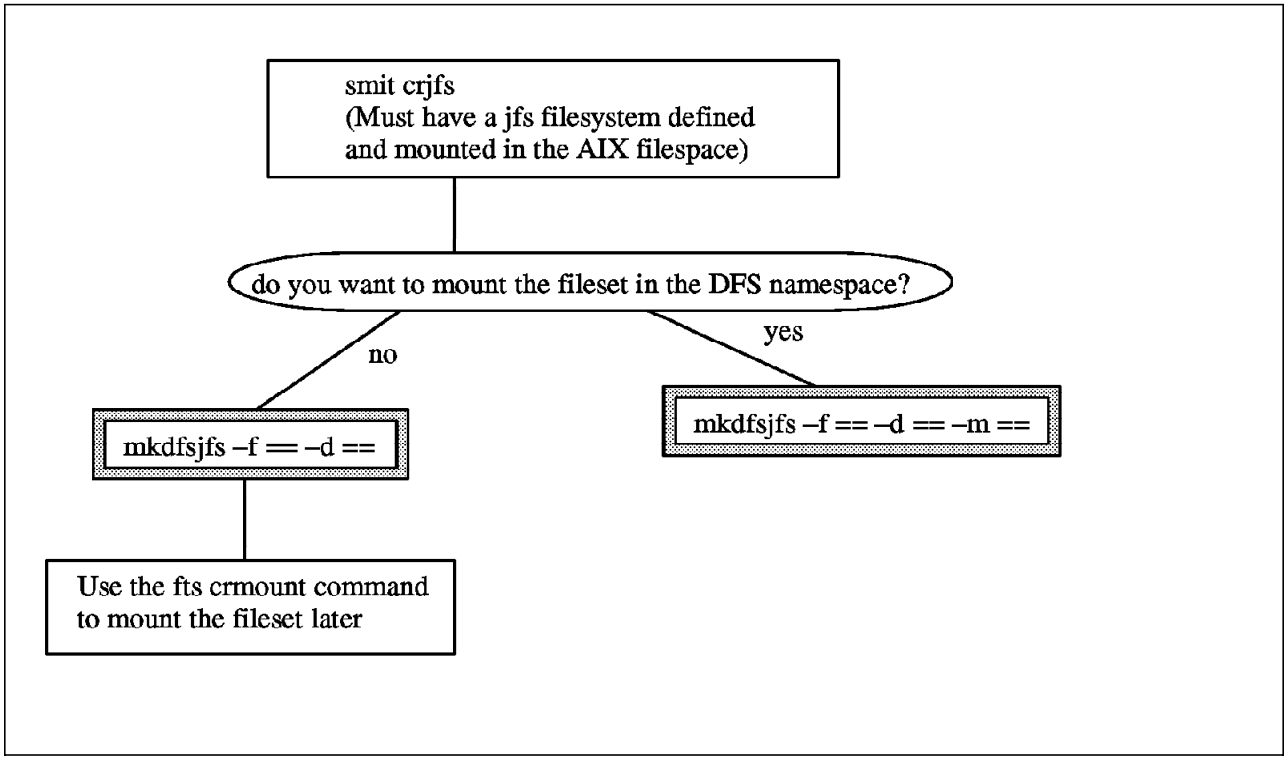


Figure 43. Adding a JFS Fileset

- `mkdfsjfs -f` *fileset_name* `-m` *DFS_mount_point* `-d` *device_name*
- For information on options, see InfoExplorer.

```
# mkdfsjfs -f myjfs -d /dev/lv05 -m /:/dir2
```

You can now see that the FLDB knows about this fileset:

```
# fts lsfldb
myjfs
        readWrite   ID 0,,22  valid
        readOnly    ID 0,,23  invalid
        backup      ID 0,,24  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner
```

```
sys4                RW       /export/jfsfs1 0:00:00 hosts/sys4     <nil>
```

You can now cd /:/dir2 (from a machine that has been configured as a DFS client).

4.4.5 Step 5: Configuring a DFS Backup System

4.4.5.1 Configuring a Backup Database machine

To configure a machine as a DFS Backup Database machine, perform the following steps:

1. As root, start SMIT with the mkdfsbkdb

   ```
   smit mkdfsbkdb
   ```

   or select the following sequence of SMIT menu options:

   a. *Communication Applications and Services*

   b. *DCE(Distributed Computing Environment)*

   c. *Configure DCE on the Local Machine*

   d. *Configure DCE Servers*

   e. *DFS Backup Database Machine*

2. For the following screen:

   - If you want this machine to use DFS′s upclient process to receive its administration lists from a System Control machine, enter the name of that machine in the “DFS System CONTROL machine to get administration lists from” field. Use its DCE name (for example, /.:/hosts/sys1). If you leave this field blank, this machine maintains its own administration lists. If this machine has already been configured as a System Control machine, this field is ignored.
   - If you specified a System Control machine in the previous step and want to alter the amount of time the upclient process waits between checks for updated administration lists, change the value of the “FREQUENCY to update administration lists (in seconds)” field. The default is 300 seconds.
   - If you specified a System Control machine in the previous step and want this machine′s upclient process to log errors that occur while it requests administration lists from the System Control machine, enter the name of log file in the “LOG file for administration list updates” field. The directory where this file will be written must already exist.
   - If this machine has already been configured as a DCE client, the other fields are already filled in like following screen.

     If they are not, use the instructions in “Configuring DCE Clients” chapter of *DCE Administration Guide* to determine the values to enter.

```
                        DFS Backup Database Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  DFS System CONTROL machine to get                  []
      administration lists from
  FREQUENCY to update administration lists (in secon []
  ds)
  LOG file for administration list updates           []
* CELL name                                          [/.../dino]
* SECURITY server                                    [sys3]
  CDS server (If in a separate network)              []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.../dino/lan-profile]



F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Undo            F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

3. Press **Enter**
4. When prompted, enter the cell administrator's password.

   It will take a few minutes to process.

NOTE

You can also configure a machine as a DFS Backup Database machine through AIX command:

 # mkdfs -s /.:/hosts/sysX dfs_bkdb

Replace sysX with the hostname of System Control machine in the DFS domain.

Please see InfoExplorer for more information on parameters needed for this command.

This procedure only configures the processes necessary for the Backup Database machine.

4.4.5.2 Configuring a Tape Coordinator Machine

Perform the following tasks once, when you initially configure a Tape Coordinator machine:

1. Attach and install the tape drives.

   You may install one or more drives on the machine. The DFS Backup System can track a maximum of 1024 drives in a cell.

2. Verify that the /opt/dcelocal/var/dfs/backup directory exists on the machine. Create the directory if it does not already exist.

   Verify that you have the write and execute permissions on the /opt/dcelocal/var/dfs/backup directory.

3. Determine the capacity of the tape in the tape drive.

The Backup System needs to know the amount of data that can be stored on the tape and the size of the *end of file (EOF)* mark. Put a blank tape in the tape drive. Issue fms -tape /dev/rmtX command that determines the size of tape. The following example was done for an 8mm tape that had a block size of 1024.

```
# fms -tape /dev/rmt0
wrote block: 142795
Finished data capacity testing - rewinding
wrote 10996 blocks, 109960filemarks
Finished filemark test
Tapecapacity is 2339586048 bytes
File marks are 196362 bytes
```

We have also tested this for a 512 byte block size on our 8mm tape.

```
# fms -tape /dev/rmt0
wrote block: 71385
Finished data capacity test - rewinding
wrote 10190 blocks, 101900filemarks
Finished filemark test
Tape capacity is 1169588224 bytes
File marks are 98392 bytes
```

You must test the fms for your tape drive and your block size as this will vary.

NOTE

The fms command takes approximately two hours for an 8mm tape with 2.3GB capacity. But the values from this command are required in the next step.

4. Create the /opt/dcelocal/var/dfs/backup/TapeConfig file on the machine with a text editor.

   Use a single command line in the file for each tape drive attached to the Tape Coordinator machine, recording:

   - The tape size of tapes to be used in the drive.
   - The *end of file (EOF) mark size* for the tape to be used in the drive.
   - The *device name* of the tape drive.
   - The TCID for the Tape Coordinator associated with the drive.

   It is recommended that you use a number 10 to 15% lower than the actual tape capacity and a number 10 to 15% higher than the actual file mark size to allow for tape variations.

   Legal values for TCID are integers from 0 to 1023. You do not have to assign the numbers in sequence. The TCID for any Tape Coordinator must be unique among all TCIDs in the local cell.

   Following is an example of the TapeConfig file for a machine with one drive. The tape size is 2 gigabytes; the EOF mark size is 200 kilobytes; the device name is /dev/rmt0; and the TCID is 0.

   ```
   2g 200k /dev/rmt0 0
   ```

5. Add the tape coordinator to the Backup Database by issuing bak addhost

   ```
   # bak addhost -tapehost sys4 -tcid 0
   Adding host 'sys4' offset 0 to tape list...done
   ```

   The syntax for the bak addhost command is as follows:

- `# bak addhost -tapehost` *BackupServer_machine* `-tcid` *tc_number*
- Replace `BackupServer_machine` with the name of your machine. Replace `tc_number` with the Tape Coordinator ID used in the previous step.
- Repeat this command for each Tape Coordinator.
- Verify the tape host has been added to the Backup Database:

```
# bak lshosts
Tape hosts:
    Host sys4, port offset 0
```

4.4.5.3 Defining Fileset Families and Fileset Family Entries

See 3.5.3, “Fileset Families” on page 64.

4.4.5.4 Defining a Dump Hierarchy of Dump Levels

See 3.5.4, “Dump Levels” on page 65.

4.4.5.5 Labeling Tapes

See 3.5.6, “Tape Labels” on page 67.

4.4.6 Step 6: Configure a DFS Client Machine

You can install a DFS client on a DCE client machine. You can also install a DFS client on the SCM, the FLDB machine or even one of the file server machines. It is helpful to configure a DFS client machine to check out whether your DFS set up is functioning correctly. You can verify that the DFS is running correctly if you can successfully cd to the /: directory. This can only be done from a machine that has been set up as a DFS client.

You can set a machine up as a DFS client either from SMIT or from the command line.

4.4.6.1 Configuring a DFS Client machine from SMIT

To configure a machine as a DFS client, perform the following steps on the machine:

1. As root, start SMIT with the `mkdceclient`

   # smit mkdceclient

   or perform the following sequence of SMIT menu options:

   a. *Communication Applications and Services*
   b. *DCE(Distributed Computing Environment)*
   c. *Configure DCE on the Local Machine*
   d. *Configure DCE Clients*

2. At the following screen:
   - In the “CLIENTS to configure” field, enter the **dfs_cl**.
   - For the “DFS CACHE on disk or memory?” field, select the location you For more information on cache, look at the 5.6, “Cache Management” on page 153.
   - For the “DFS cache SIZE (in kilobytes)” field, enter the size to be used for the DFS client cache. For an on-disk cache, this value should not exceed 90% of the disk space in the file system where the cache is to be located. For an in-memory cache, this value should not exceed 25% of the machine’s available memory.

- In the “DFS cache DIRECTORY (if on disk)” field, specify the directory where the DFS client cache files should be kept. If you selected memory in the previous step, this field is ignored. It is strongly recommended that you create a separate JFS file system for the DFS client cache if you are keeping it on disk. The default directory is /var/dce/dfs/cache, and the default cache size is 10MB.
- If this machine has already been configured as a DCE client, the other fields are already filled in like the following screen. If they are not, use the instructions in “Configuring DCE Clients” chapter of the *AIX DCE Administration Guide* to determine the values to enter.

```
                                DCE Client

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* CLIENTS to configure                               [dfs_cl]                +
* CELL name                                          [/.../dino]
* SECURITY server                                    [sys3]
  CDS server (If in a separate network)              []
* Cell ADMINISTRATOR's account                       [cell_admin]
* LAN PROFILE                                        [/.../dino/lan-profile]
* DFS CACHE on disk or memory?                       [disk]                  +
* DFS cache SIZE (in kilobytes)                      [10000]
* DFS cache DIRECTORY (if on disk)                   [/opt/dcelocal/var/adm/>


F1=Help          F2=Refresh       F3=Cancel        F4=List
F5=Undo          F6=Command       F7=Edit          F8=Image
F9=Shell         F10=Exit         Enter=Do
```

3. Press **Enter**
4. When prompted, enter the cell administrator’s password. At this time, the machine is configured as a DFS Client machine.

4.4.6.2 Configuring a DFS Client from the Command Line

You can also configure a machine as a DFS Client machine through the AIX command:

```
# mkdfs dfs_cl
```

At this time, we don’t specify any options here because we want to use all of the defaults for a DFS client which are:

- *The cache is on disk.*
- *The cache directory is /var/dce/dfs/adm/cache*
- *The cache size is 10MB.*

If you do not want to use all of the default values, use the `-c` flag to specify the directory, the `-s` flag to specify the size, and the `-m` flag to have the cache in memory.

Note

We recommend you create a new AIX JFS file system to hold the DFS cache files that are located in /var/dce/dfs/adm/cache. This is because you want to guarantee that you will have 10MB of space for the cache that you have allocated. You do not want to have some other process take up this valuable room.

The /var directory is used in AIX to house temporary files that are used by other subsystems. These other subsystems can affect the amount of filespace that is left in /var.

File System Size For the Cache

The file system size that is used for the cache must be 15% larger than the size of the cache itself. For a cache size of 10MB, your JFS file system size must be 11.8MB.

Token State Recovery Can Take a Few Minutes...

Note that when you cd /: for the first time after setting up the root.dfs fileset, it could possibly hang or even return a TSR related error. It may take on the order of minutes if the file server is in Token State Recovery (TSR) mode for the cd /: to be successful.

Chapter 5. DFS Administration Tasks

System administration of DFS requires knowledge about DCE and DFS components. It is a fairly complicated task and should be handled carefully by a trained person. If one component such as the directory service in a distributed environment does not function correctly, everything may be out of service. In this chapter we will discuss how principals and groups receive permission to administer the DFS. This is done with administrative domains and lists. Next, we will discuss the utilities that are available to the administrators. Finally, we will talk about the actual process of system administration.

The system administration for DFS can be broken down into the following tasks:

- Fileset management
- Monitoring and Controlling the DFS Servers
- Backup and Restore of DFS filesets
- Cache Management
- Security Management

The security management task will be presented in Chapter 6, “Security” on page 161.

Due to the many commands and options involved in DFS administration we cannot explain them all in this document. Please refer to IBM InfoExplorer or the *DFS Administration Guide* for further information. We will provide basic information on these subjects and give plenty of examples. The examples shown in this chapter assume a cell layout as outlined in Figure 44 on page 110 and the example commands were issued mostly from host /.../dino/hosts/sys3.

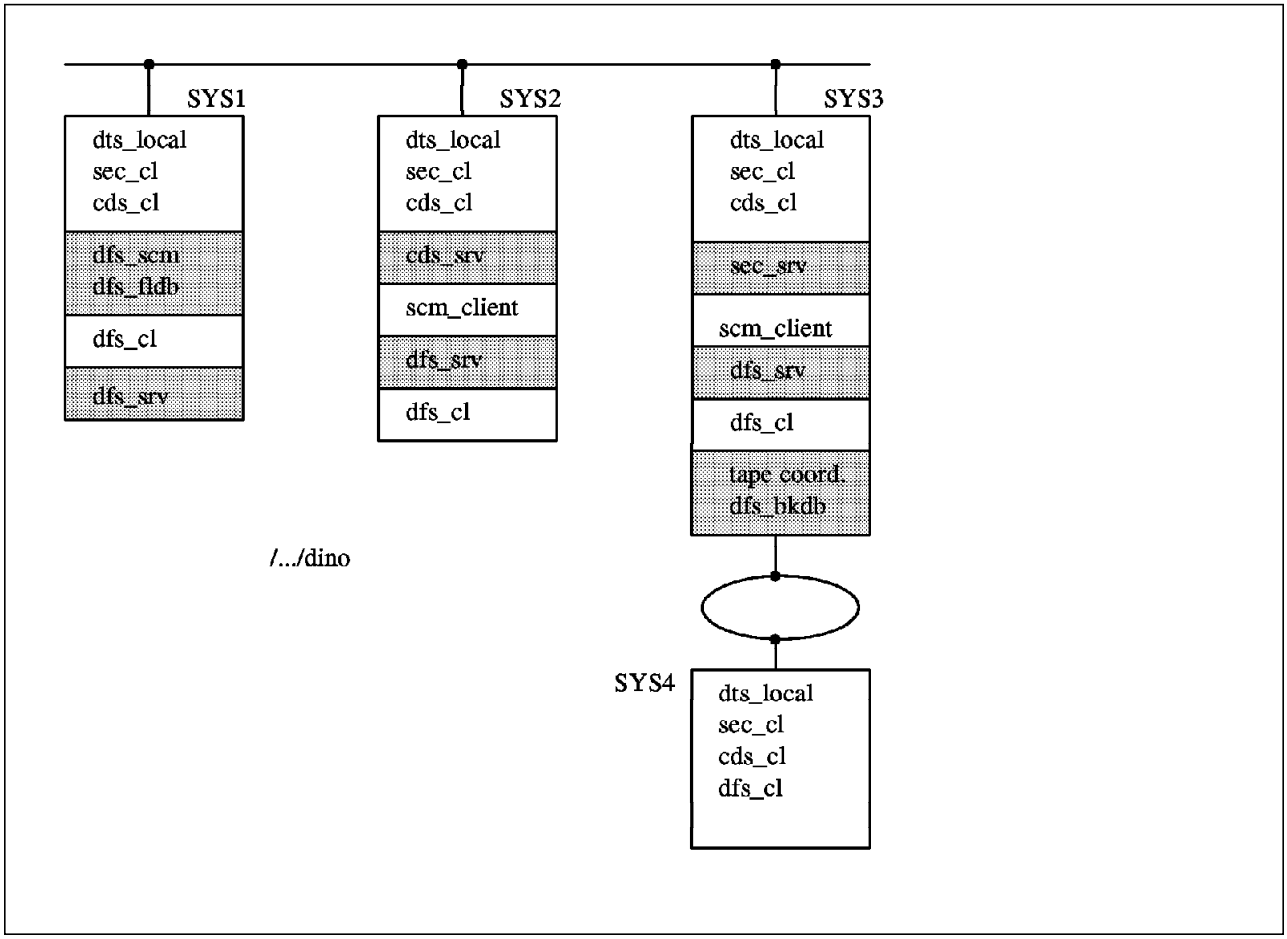


Figure 44. Our DCE Layout -- the /.../dino Cell

5.1 Administrative Domains and Lists

DCE-DFS provides the concept of administrative domains. A DFS administrative domain is a collection of associated DFS server machines which belong to the same cell and which are setup to be administered as a single unit. The purpose of DFS administrative domains is to further sub-divide the cell into smaller units which are easier to administer. All members of the DFS administrative domain must belong to the same cell although each machine can be a member of several DFS administrative domains concurrently. In addition to simplifying the administration of a cell, DFS administrative domains introduce methods for fine granularity of granting access rights and a great flexibility for administration in general.

A DFS administrative domain uses DFS administrative lists to control which principals have privileges to issue requests to specific server processes.

There are several DFS administrative lists for different tasks and you need to understand which list must be available on particular DFS servers. To have the permission to administer the complete DFS administrative domain you (the DCE principal) must be listed in all of them. The admin lists grant permission to selected principals or groups for certain administration tasks.

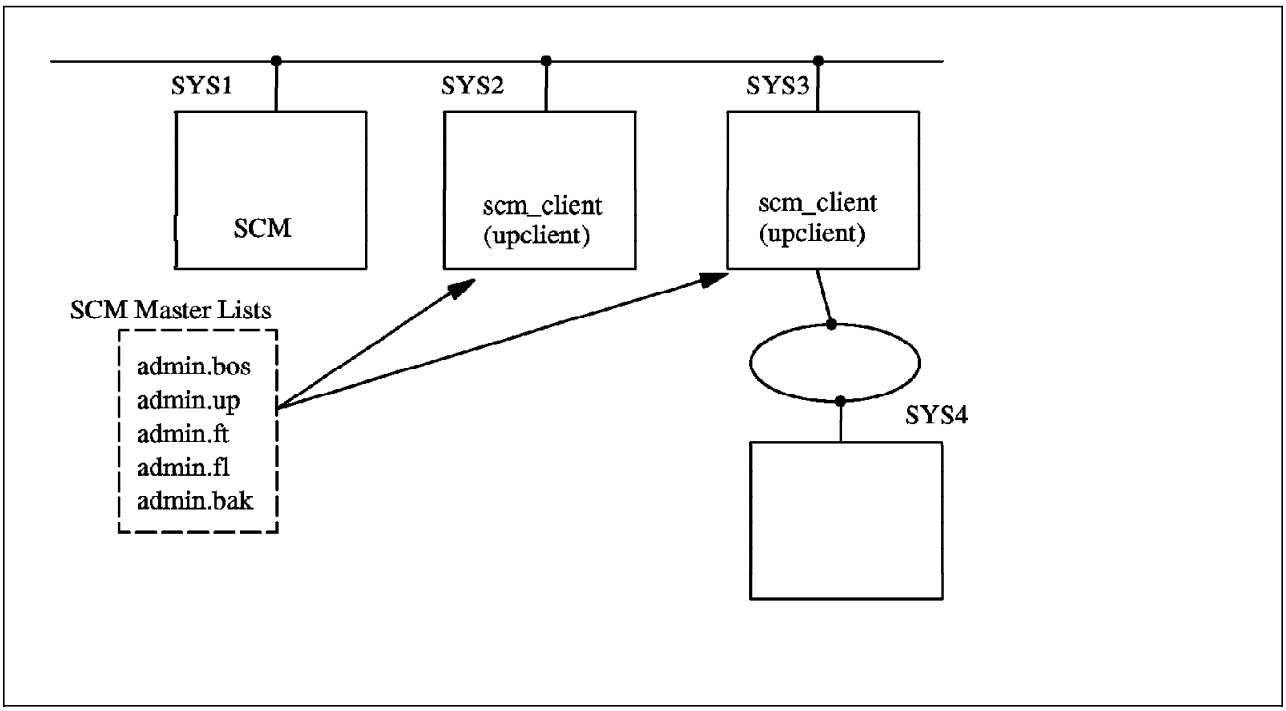


Figure 45. Administration Lists and DFS

An easy method of handling this issue is to define a group in the DCE security registry, place this group into the DFS administrative lists and then add all administrators to this group. You can use the `rgy_edit` command to create groups and then use the `bos addadmin` command to place the groups on administrative lists. In fact, a similar setup is provided as the default. A group with the name subsys/dce/dfs-admin has been created at DFS configuration time and this group has been added to all DFS administrative lists. The cell_admin principal is a member of this group and has therefore by default all privileges to execute the administrative tasks. You can add other DCE principals to the subsys/dce/dfs-admin group to provide them with equivalent rights. This is done with the `rgy_edit` command.

Note

The DFS administrative lists are stored locally in the /opt/dcelocal/var/dfs directory (there is a symbolic link from /opt/dcelocal/var ---> /var/dce, which makes the DFS administrative lists also available from /var/dce/dfs).

The files are in binary format and you cannot use vi or pg to edit or list them, only `bos` commands are allowed to work with DFS administrative lists. They are stored on the local system, they control privileges for DFS server processes which run on that same system. For example, if you issue a command on machine sys3 which queries data on sys2, then you need the correct entries in the DFS administrative lists on sys2. If you move a fileset from sys3 to sys2 then you need entries in the DFS administrative lists in both systems.

To make the process of controlling which user or group has to be included into which DFS administrative list easier, you should use the DFS administrative lists of the DFS System Control Machine as the master lists. With the help of the upserver and the upclient processes you distribute the DFS administrative lists from the DFS System Control Machine to all other DFS server systems in the

same administrative domain. The interval of distributing updates of these lists defaults to five minutes, but can be altered with the -time option of the upclient process.

Table 9. Administrative Lists

Name	Server Process	Permissions Granted	Entries in our Lab Setup
admin.bos	BOS Server (bosserver)	Manage DFS Server Processes, maintain key files and admin lists	group: subsys/dce/dfs-admin (default)
admin.up	Update Server (upserver)	Allow upclient processes to retrieve files	group: subsys/dce/dfs-admin (default)
admin.ft	Fileset Server (ftserver)	To administer filesets	users: hosts/sys1/dfs-admin; hosts/sys2/dfs-admin; hosts/sys3/dfs-admin; hosts/sys4/dfs-admin; group: subsys/dce/dfs-admin (default)
admin.fl	FLDB Server (flserver)	To maintain FLDB	users: hosts/sys1/dfs-admin group: subsys/dce/dfs-admin (default)
admin.bak	Backup Server (bakserver)	To maintain Backup Database	users: hosts/sys1/dfs-admin group: subsys/dce/dfs-admin (default)

As you see from the previous table, the group subsys/dce/dfs-admin is included in every DFS administrative list. This is the default setup. The principal cell_admin is a member of this group. Therefore if you login to your DCE cell with principal cell_admin, you have access rights and privileges for all DFS related server processes.

Warning

A principal or group must be created in the DCE registry before it can be used in DFS administrative lists and must be deleted from the DFS administrative lists before it is removed from the DCE registry.

Here is how to check the members of a group and how to add additional members to this group with the rgy_edit command.

```
# rgy_edit
Current site is: registry server at /.../dino/subsys/dce/sec/master
rgy_edit=> do group
Domain changed to: group
rgy_edit=> view -m
     ...................
     ...................
     ...................
subsys/dce/dfs-admin                          103
  5 members
    cell_admin, hosts/sys1/dfs-server, hosts/sys3/dfs-server
    hosts/sys4/dfs-server, hosts/sys2/dfs-server
     ...................
     ...................
     ...................
rgy_edit=> m
Enter group name: subsys/dce/dfs-admin
```

```
Enter name to add: peter
Enter name to add:
Enter name to remove:
Enter group name:
rgy_edit=>
```

The principal peter must have been established beforehand, but now you can login to dce with principal peter and he has privileges to use all administration tasks that the dfs-admin group has privileges for.

See Appendix B, “Multiple Administrative Domains” on page 229 for information on how to setup several administrative domains, which is useful if you have very large cells.

You may need to disable the authorization checking temporarily for emergency system management. (For example, if your key files become corrupted or out of sync.) This can be done by creating the zero length file /var/dce/dfs/NoAuth on the DFS file server machine. There is a command to disable authorization checking remotely (`bos setauth -server /.:/hosts/sys3 off`) but your file server and the DCE Security Service must be functional. You can now issue most commands without authorization checking.

Warning

Do not disable authorization checking longer than necessary because it allows any user to execute DFS commands and is therefore a security exposure.

5.2 DFS Administration Utilities

DFS provides a series of commands for administration. They are categorized into the following command suites:

bos The bos command suite is used to monitor and control DFS server processes

cm The cm command suite is used to customize and control the DFS clients.

bak The bak command suite is used to operate the DFS Backup System.

fts The fts command suite is used to manage filesets.

Each of these are called a command suite because each has subcommands that offer a range of function. In fact it is difficult to remember all options on these command suites. The command suites provide help messages. Simply provide the word *help* as a parameter to the commands, such as:

```
# bos -help
# fts crmount -help
```

The first example shows the complete range of sub-commands in a command suite. The second example will provide the exact sequence of parameters for a particular command. (Note that the output of both of these commands have not been shown here.)

5.3 Aggregate and Fileset Management

As we know from previous chapters, there are two types of filesets available with DFS: LFS filesets and non-LFS filesets. DFS aggregates can either house one non-LFS fileset or multiple LFS filesets. Information about the filesets is stored in the Fileset Location Database (FLDB) and also in fileset headers at each site containing a copy of the fileset. It is important that the FLDB and the local fileset header information are always kept in synchronization. This section will describe how to obtain information about filesets, how to create, delete, move, rename, lock, unlock, replicate, and clone filesets. It will also explain how to set and change quotas on filesets.

5.3.1 Creating DFS Aggregates and Filesets

DFS filesets can be of type LFS or non-LFS. The procedure of creating these different types of filesets is not the same, but similar.

5.3.1.1 Creating DCE LFS Aggregates and Filesets

Let us quickly get an overview of the steps involved creating a DCE LFS fileset before we describe the procedure in more detail. Creating an DCE LFS Fileset requires the following steps if you want to create this fileset on a new aggregate:

1. Make logical Volume (`mklv`)
2. Create an aggregate (`newaggr`)
3. Edit /var/dce/dfs/dfstab
4. Export the aggregate (`dfsexport`)
5. Create a fileset (`fts create`)
6. Create a mount point (`fts crmount`)

If you want to create the fileset on an existing aggregate, only steps 5 and 6 are necessary.

The IBM command to perform step 3 through step 6 is `mkdfslfs`, which is easier to use, but it is helpful to see all individual steps.

You must use the `newaggr` command to initialize and format an aggregate on a logical volume in AIX Version 3 before you can create filesets. This command creates the metadata structure used for ACLs and logging. Use the following command to create a new logical volume with the name *lfsusers* and a size of 4 logical partitions (4MB each). Then use `newaggr` to initialize a new aggregate in this logical volume.

```
# mklv -y lfsusers  -t lfs rootvg 4
# newaggr -aggregate lfsusers -blocksize 8192 -fragsize 1024 -overwrite
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 2047
*** Defaulting to 20 log blocks    (maximum of 1 concurrent transactions).
/dev/rlfsusers: Marked as not a BSD file system any more.
Done.  /dev/rlfsusers is now an Episode aggregate.
```

The blocksize and fragsize parameters are required. They specify the number of bytes in each DCE LFS block and the number of bytes in each fragment. The block size must be a power of 2 between 4096 and 65536. The maximum number of blocks that can be addressed in an DCE LFS fileset is 2**31 thus defining the largest theoretical file size as 2**31 times 65536 bytes which is 2**47 bytes.

(Actually, the maximum file size is constrained by other factors as well.) The fragment size must be a power of two between 1024 and blocksize.

Also note the usage of the -overwrite option on the newaggr command. This specifies that an existing file system may be overwritten.

Note

Once you make a logical volume and create the aggregate on it, the remaining steps could be performed with the IBM command:

 # mkdfslfs -f user1.fs -m /:/user1 -d /dev/lfsusers -n lfsusers

The following steps could be used in place of the mkdfslfs command that is shown above:

Export the aggregate to the DFS filespace. This is done by editing the dfstab file and then issuing the dfsexport command.

```
# vi /var/dce/dfs/dfstab
  # blkdev        aggname     aggtype    aggid   [UFS fsid]
  /dev/lfsusers  lfsusers   lfs        1
#
#
# dfsexport -aggregate lfsusers
```

The dfsexport command will export the aggregate and put an entry into the dfsatab file, residing in the same directory. The dfsatab file, which can be viewed but should not be edited, always holds the currently exported aggregates.

The next step involves creating one or more filesets on the aggregate. An example is outlined here creating a fileset with the name user1 on the file server /.:/hosts/sys3 in the aggregate lfsusers.

```
# fts create -ftname user1.fs -server /.:/hosts/sys3 -aggregate lfsusers
        readWrite   ID 0,,275  valid
        readOnly    ID 0,,276  invalid
        backup      ID 0,,277  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

192.100.100.3       RW       lfsusers 0:00:00 hosts/sys3     <nil>

Fileset 0,,275 created on aggregate lfsusers of /.:/hosts/sys3
```

The fileset has now been created on the File Server /.:/hosts/sys3 and has been assigned the ID of 275 by the system. You can view it with the fts lsheader command or with the fts lsfldb command. However, before you can actually use the fileset, you must create a mount point for it.

```
# fts crmount -dir /:/user1 -fileset user1.fs
```

The fileset is created and established in the FLDB. It is now mounted and it is ready to use. You may want to change ownership and group with chown and chgrp and then use acl_edit on the mount point to set initial access rights before you allow users to access the directory. For information on ACLs see Chapter 6, “Security” on page 161. You might also want to change the fileset quota. The default value is 5000KB. See the fts setquota command in IBM InfoExplorer.

5.3.1.2 Creating Non-LFS Filesets

The procedure to define a non-LFS fileset is different from creating a DCE LFS fileset. Defining a non-LFS Fileset requires the following steps:

1. Create a JFS file system (crfs)
2. Mount the JFS file system (mount)
3. Create an entry in the FLDB (fts crfldbentry)
4. Edit /var/dce/dfs/dfstab
5. Export /var/dce/dfs/dfstab (dfsexport)
6. Create a mount point (fts crmount)

The IBM added value command to perform step 3 to step 6 is mkdfsjfs.

We will refer to JFS file systems that have been DFS-exported as being non-LFS aggregates. These non-LFS aggregates cannot contain multiple filesets or support the DCE ACLs. To create a non-LFS fileset, perform the following steps. First, create a JFS file system with SMIT or crfs and mount it (or you can use an existing locally mounted JFS file system).

```
# crfs -v jfs -g rootvg -m /export/user2 -a size=10000
# mount /export/user2
```

Note

The remaining steps 3-6 can be performed with the IBM command:

 # mkdfsjfs -f user2 -m /:/user2 -d /dev/lvXX

If you do not use the IBM-only mkdfsjfs command that was mentioned above and instead you want to use the standard DCE DFS commands, then you must create an entry in the FLDB with the fts crfldbentry command. The aggregate ID should be an unused ID and not show up in the dfstab file so far. So look into /var/dce/dfs/dfstab before you select the aggregate ID on the following command.

```
# fts crfldbentry -ftname user2 -server /.:/hosts/sys3 -aggrid 9
        readWrite   ID 0,,281  valid
        readOnly    ID 0,,282  invalid
        backup      ID 0,,283  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

192.100.100.3       RW       9       0:00:00 hosts/sys3     <nil>

FLDB entry created for fileset user2 (0,,281) on aggregate 9 of /.:/hosts/sys3
```

In the next step you use a text editor to edit one line in the /var/dce/dfs/dfstab file and define the non-LFS aggregate. The /var/dce/dfs/dfstab file is used by the dfsexport command to export the aggregate that you specify. You get the block device by listing all logical volumes. This is done with the AIX lsvg -l rootvg command. The aggregate ID and the fileset ID are available from the output of the crfldbentry command.

```
# vi /var/dce/dfs/dfstab
  # blkdev        aggname         aggtype    aggid   [UFS fsid]
  /dev/lv04       /export/user2   ufs        9       0,,281
#
#
# dfsexport -aggregate /export/user2
```

Finally, a mount point has to be created for the new non-LFS fileset.

```
# fts crmount -dir /:/user2 -fileset user2
```

The non-LFS fileset is now mounted in the DFS file space and is ready to use. You may want to change user and group ownership with `chown` and `chgrp` commands and set the mode bits with the `chmod` command.

5.3.2 Deleting Filesets and Aggregates

All three types of filesets, readWrite, readOnly and backup can be removed individually. When you remove the readWrite version, the master copy of the fileset is gone and the backup version of the fileset is automatically deleted. You cannot delete a readWrite fileset if a readOnly version of a fileset also exists. (See the `fts rmsite` command in IBM InfoExplorer for information on how to remove a readOnly fileset.)

5.3.2.1 Deleting a DCE LFS Fileset

The steps for removing an LFS fileset are the following:

1. Remove the mount point (`fts delmount`)
2. Delete the fileset (`fts delete`)

Note

The IBM command to perform this is `rmdfslfs`. This can be used as follows:

```
rmdfslfs -m /:/user1 -f user1.fs
```

First, you should delete the mount point for the LFS fileset.

```
# fts delmount -dir /:/user1
```

Then use the following command to delete the fileset:

```
# fts delete -fileset user1.fs -s /.:/hosts/sys3/ -aggregate lfsusers
```

This command will delete the entry from the FLDB and also remove the fileset. The `fts delete` command can only be used when the entry in the FLDB and the fileset exists. If you need to delete only one of these, then use the `fts delfldbentry` to delete the entry from the FLDB or the ″fts zap″ command to delete the fileset.

5.3.2.2 Deleting a DCE LFS Aggregate

The steps for removing an LFS aggregate are the following:

1. Unexport the aggregate (`dfsexport -detach`)
2. Remove the entry from dfstab file
3. Remove the logical volume that the aggregate was defined on

Note

The IBM command to perform this is rmdfslfs -F -n <aggr_name>. (This will do the dfsexport -detach and remove the aggregate from the dfstab file. You would still have to remove the logical volume.)

Note that you should only delete an aggregate when you have removed all the LFS filesets from that aggregate and you will no longer need the aggregate. First you must unexport the aggregate.

```
# dfsexport -aggregate lfsusers -detach
```

Then remove the entry from the /var/dce/dfs/dfstab file with an editor.

Finally, remove the logical volume by using smitty rmlv or with:

```
# rmlv -f  name_of_lv
```

5.3.2.3 Deleting a Non-LFS Fileset

The procedure for removing a non-LFS fileset is the following:

1. Remove the mount point (fts delmount)
2. Delete the entry from FLDB (fts delfldbentry)
3. Unexport the non-LFS aggregate (dfsexport -detach)
4. Edit the /var/dce/dfs/dfstab file to remove the aggregate entry
5. Unmount the JFS file system (umount)
6. Remove the file system (rmfs)

(Note that the fts delete command can not be used with non-LFS fileset.)

Note

The IBM command to perform this is rmdfsjfs. This can be done as follows:

rmdfsjfs -F -m <mountpointname -f <filesetname>
or
rmdfsjfs -F -m <mountpointname -d <devicename>

This will remove the fileset (filesystem) from the DFS and delete the mount point.

First remove the mount point from the DFS filespace.

```
# fts delmount -dir /:/user2
```

Next, delete the entry from the FLDB with the following command:

```
# fts delfldbentry -fileset user2
```

The aggregate must then be unexported. This is done as follows:

```
# dfsexport -aggregate /export/user2 -detach
```

Next, delete the entry from the dfstab. Use an editor on the file /var/dce/dfs/dfstab and remove the appropriate entry.

You can now unmount the JFS local file system now and destroy it.

```
# umount /export/user2
# rmfs /export/user2
```

5.3.3 Listing Fileset Information

The following commands can be used to obtain information about filesets:

- `fts lsfldb` - show information from FLDB
- `fts lsft` - show fileset header information
- `fts lsheader` - show information from fileset header
- `fts lsquota` - show information about quotas
- `fts lsmount` - show information about mount points
- `fts lsreplica` - show information about replicas
- `fts lserverentry` - show server information from FLDB
- `fts lsaggr` - lists all aggregates on a server
- `fts aggrinfo` - show disk usage information about aggregates on a server

Not all of these commands and all options to these commands can be explained in this document. They are well documented in IBM InfoExplorer and also in the *DFS Administration Guide*.

Very commonly used are the `fts lsfldb` and the `fts lsheader` commands to get current information about DCE LFS and non-LFS filesets. See an example of the `fts lsfldb` command, which queries the FLDB for requesting information about filesets.

```
# fts lsfldb user3.peter
        readWrite   ID 0,,220  valid
        readOnly    ID 0,,221  invalid
        backup      ID 0,,222  valid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

192.100.100.3       RW,BK    lfspeter 0:00:00 hosts/sys3     <nil>
```

As we can see from this example, the `fts lsfldb` displays three lines showing the readWrite, readOnly, and backup version of the fileset user3.peter with their IDs and their current status (invalid means the fileset does not exist). The next line indicates the number of sites where this fileset resides. In our case, it exists only at one site. And the last line shows:

1. the Network address of the file server system
2. the Flags RW (read-write) RO (read-only) or BK (backup)
3. the name of the aggregate.
4. the MaxSiteAge parameter (This is the maximum amount of time that a replica can be out of date for a RO fileset. For a RW fileset this is set to 0:00:00)
5. the Principal name of the File Server
6. the Name of the group that owns the server entry (`<nil>` in our case)

Now let us also look at an example of the output of the fts lsheader command, which displays fileset header information without querying the FLDB.

```
# fts lsheader -s /.:/hosts/sys3 -aggregate lfspeter
Total filesets on server /.:/hosts/sys3 aggregate lfspeter (id 1): 4
user3.peter              0,,220 RW      8 K alloc     17 K quota On-line
user3.peter.backup       0,,222 BK     17 K alloc     17 K quota On-line
user3.peter.re           0,,253 RW      8 K alloc      9 K quota On-line
user3.peter.re.backup    0,,255 BK      9 K alloc      9 K quota On-line
Total filesets on-line 4; total off-line 0; total busy 0
```

Note

Non-LFS filesets actually have no fileset header, but the fts lsheader can still be used to display some information.

The output shows information from the host sys3 about the filesets in the aggregate named lfspeter. Each fileset in this aggregate is listed with a name, the fileset ID, the fileset type, its current size in kilobytes, its quota usage in kilobytes and a status indicating whether the fileset is on-line or off-line. There is a -long option on the fts lsheader command which displays more details. Use IBM InfoExplorer to learn about these details.

A combination of fileset header information and FLDB information is available from the fts lsft command. This command is useful to display information about a fileset when all you know is it's pathname for the file or directory that you are interested in.

```
# fts lsft -path /:/user3/peter
user3.peter 0,,220 RW LFS     states 0x10010005 On-line
    192.100.100.3, aggregate lfspeter (ID 1)
    Parent 0,,0       Clone 0,,221     Backup 0,,222
    llBack 0,,222     llFwd 0,,0       Version 0,,322
    4294967295 K alloc limit;          8 K alloc usage
          5000 K quota limit;         17 K quota usage
    Creation Tue Jul  6 13:50:19 1993
    Last Update Mon Jul 12 05:30:35 1993


user3.peter
        readWrite   ID 0,,220  valid
        readOnly    ID 0,,221  invalid
        backup      ID 0,,222  valid
number of sites: 1
   server           flags     aggr   siteAge principal      owner

192.100.100.3       RW,BK    lfspeter 0:00:00 hosts/sys3     <nil>
```

As you can see from the previous examples, the output of these three commands (fts lsfldb, fts lsheader, and fts lsft) provides you with information about location, ID numbers, quotas, and many other details about filesets.

5.3.4 Moving Filesets

Consider moving filesets to other aggregates or sites if the current machine or disk must be repaired or if the aggregate becomes full. Moving filesets in DFS has some restrictions:

- Only DCE LFS filesets can be moved. Non-LFS filesets cannot be moved with the `fts move` command. See 5.3.4.2, “Moving Non-LFS Filesets” to see how non-lfs filesets are moved.
- Only readWrite versions of filesets can be moved.
- Filesets can be moved between sites in the same cell only
- Filesets cannot be mounted locally if you want to move them (you also need to unmount backup versions).
- When you move the readWrite version of a fileset, the backup copy is deleted automatically.

DCE LFS Fileset Benefit

A benefit of using DCE LFS filesets is that they can be moved without taking them off-line. The end user may not realize that they are being moved. After having moved a fileset, if you want the backup version of your fileset to be re-established immediately, then you must run the `fts clone` command at the new site to create a new version of the backup.

There is no need to change the mount point of a fileset when it is moved.

5.3.4.1 Moving DCE LFS Filesets

Use the following command to move the fileset user3.andrea from aggregate lfsniki from system sys3 to aggregate lfspeter on system sys3.

```
# fts move -fileset user3.andrea -fromserver /.:/hosts/sys3 -fromaggr lfsniki \
-toserver /.:/hosts/sys3 -toaggregate lfspeter
```

5.3.4.2 Moving Non-LFS Filesets

As we said already, non-LFS filesets cannot be moved with the `fts move` command, but there is a way to move non-LFS filesets. Here are the steps for moving a non-LFS fileset called user3.tom on the aggregate /export/tom to a remote site:

1. Eliminate the mount point:

   ```
   # fts delmount -dir /:/user3/tom
   ```

2. Delete the entry from the FLDB with the following command:

   ```
   # fts delfldbentry -fileset user3.tom
   ```

3. Detach the aggregate. Since this is a non-LFS fileset there is only one fileset in the aggregate and this command does not impact other filesets.

   ```
   # dfsexport -aggregate /export/tom -detach
   ```

4. Eliminate the entry from the /var/dce/dfs/dfstab table. Use any editor to perform this step.
5. You must now copy the AIX Journaled File System to the remote site, which can be achieved via various methods, such as ftp, rcp, backup/restore, and others. If you want to move the non-LFS fileset on the same site, you may simply use the AIX copy logical volume command (`cplv`).

6. On the new site, create an entry in the FLDB. Choose an aggregate ID which has not yet been used by first looking at the /var/dce/dfs/dfstab table for a list of the used IDs.

   # fts crfldbentry -ftname user3.tom -s /.:hosts/sys1 -aggrid 7

   Write down the ID of the fileset from the output of the fts crfldbentry command.

7. Edit the /var/dce/dfs/dfstab table with the new entry:

```
# vi /var/dce/dfs/dfstab
  # blkdev        aggname         aggtype    aggid   [UFS fsid]
  /dev/lv04       /export/tom     ufs        7       0,,281
#
```

8. Export the aggregate:

   # dfsexport -aggregate /export/tom

9. Create a new mount point

   # fts crmount -dir /:/user3/tom -fileset user3.tom

Steps 6 through 9 can also be done using the IBM mkdfsjfs command on sys1 as follows:

mkdfsjfs -f user3.tom -d /dev/lv04 -m /:/user3/tom

Note that this procedure involves taking the fs off-line, moving it and bringing it back on-line again. This would not be transparent to an end user.

5.3.5 Renaming Filesets

To change the name of LFS filesets and non-LFS filesets use the fts rename command. This will implicitly also change the name of the readOnly version and the backup version. The mount point must be changed manually by the administrator. This is because the mount point references the name of the fileset, which can be seen with the fts lsmount command. Use the following command sequence to rename a fileset:

```
# fts rename -oldname user3.peter -newname user3.richard
# fts delmount -dir /:/user3/peter
# fts crmount -dir /:/user3/richard -fileset user3.richard
```

Note that in order to do this, the issuer must be listed in the admin.ft file on the machine that user2.peter resides. The issuer must also be listed in the admin.fl file on all FLDB machines.

5.3.6 Listing Logical Volumes, Aggregates and Filesets

You may want to determine what logical volume names, aggregates and filesets are being used in a cell. Please refer to Figure 46 on page 123 during the following discussion.

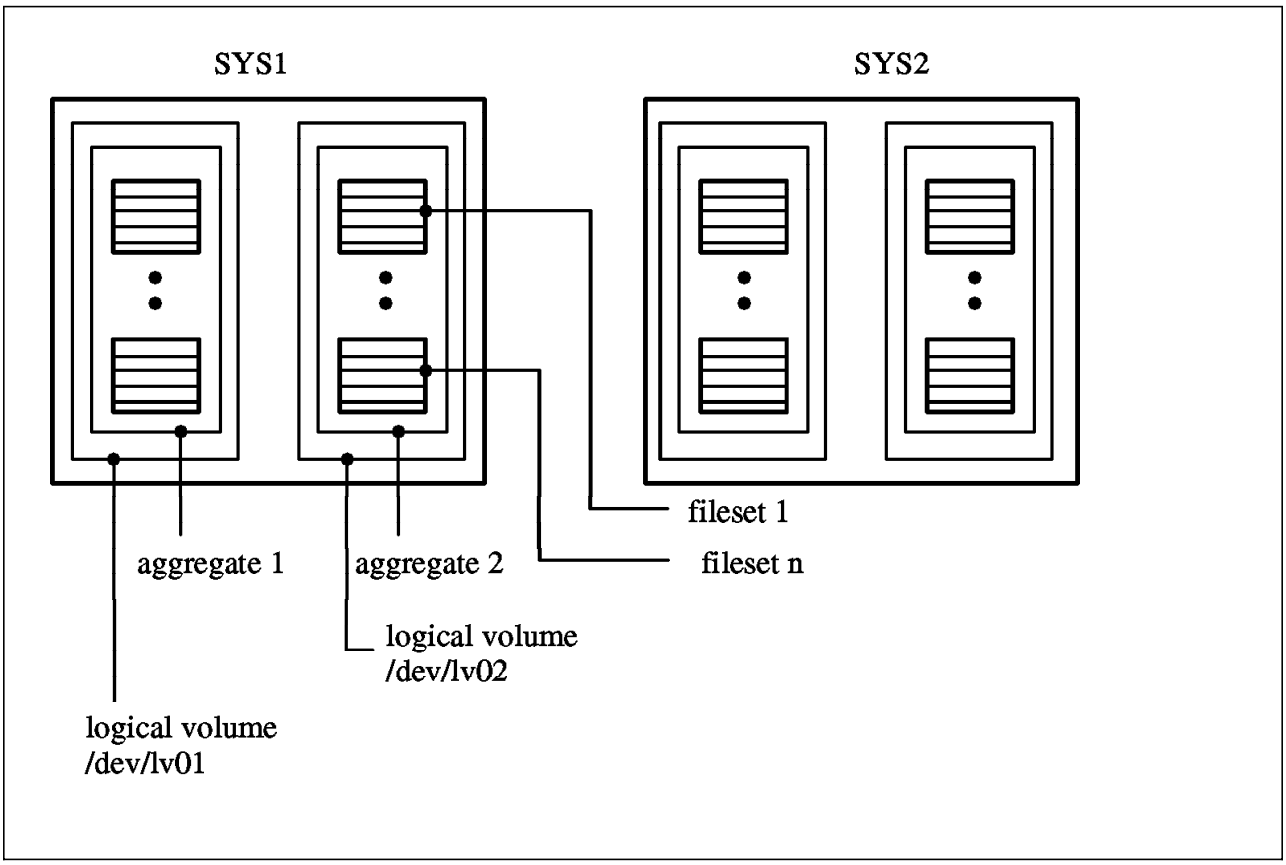


Figure 46. Logical Volumes, Aggregates and Filesets Relationship for DCE LFS

5.3.6.1 How to Find Logical Volumes

Use the following command to show the logical volumes on a particular server:

```
# lsvg -l rootvg
```

5.3.6.2 How to Find Aggregates

From Figure 46, it is apparent that you have to specify the name of the server that you want to list the aggregates on. You can use the following command to do this:

```
# fts lsaggr -server sys1
```

You can also use the `fts aggrinfo` command to display information about aggregates that are resident on a particular server.

5.3.6.3 How to Find Filesets

There are different methods to find a particular fileset. The method that you would used depends upon which information you have. From Figure 46 you can find all the filesets on a particular server by using:

```
# fts lsheader -server sys1
```

You could also look at the FLDB to determine the filesets on a particular server:

```
# fts lsfldb -server sys1
```

If we wanted to go directly to the FLDB machine and list *all* of the filesets in the cell, we could use:

```
# fts lsfldb
```

You can even list information pertaining to the location of a file or directory. The following command shows the cell, the hostname and the fileset of where a given file is located.

```
# cm whereis myfile
The file 'myfile' resides in the cell 'pursig', in fileset 'craig.lfs',
on host sys4_tr.
```

5.3.7 Increasing the Size of a DCE LFS Aggregate

In DCE DFS where filesets are contained in aggregates and aggregates are contained in logical volumes it is sometimes necessary to increase the size of an aggregate. It is useful to first obtain information about the size of logical volumes, aggregates and space taken by filesets before changing any of these objects. Look at the following figure to become acquainted with the relation of filesets, aggregates and logical volumes.

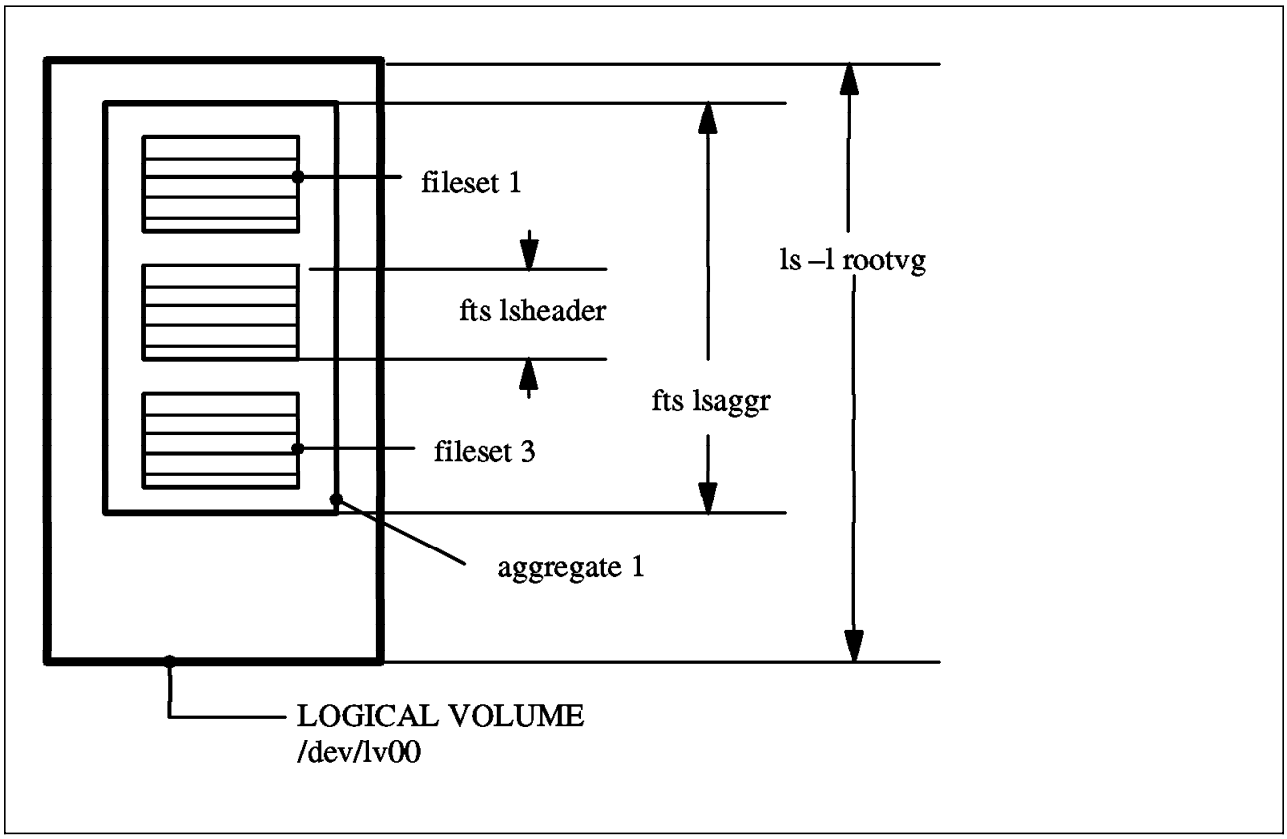


Figure 47. Determining Sizes of Logical Volumes, Aggregates and Filesets

You must increase logical volumes so that they provide enough space before you can increase aggregates and you must increase aggregates before filesets can grow or before more filesets can be created. We will look at how you determine how much of an aggregate is being used. We then will look at how you determine and extend the size of logical volumes and aggregates.

5.3.7.1 How to Determine How Much of an Aggregate is Used

By supplying the name of a host and an aggregate on that host, you can obtain the total amount of disk space on the aggregate and also the amount of disk space still available on the aggregate.

```
# fts aggrinfo -s /.:/hosts/sys3 -aggregate lfsuser3
LFS aggregate lfsuser3 (/dev/lfsuser3): 10827 K free out of total 15992
```

The output shows that there are 10827 blocks (1 kilobyte each) free out of the total of 15992 blocks (1 kilobyte each).

5.3.7.2 How to Determine and Extend the Size of a Logical Volume

Use one of the following commands to obtain the size of the logical volume where your aggregate resides. Notice, that the -noaction parameter on the growaggr command can always be used to find information without actually performing the action. The growaggr command can only be used for DCE LFS aggregates.

```
# lslv lfsuser3
# lsvg -l rootvg
# df
# growaggr -aggregate lfsuser3 -noaction
aggregate lfsuser3 holds 12288 (1024B) blocks, 12288 assigned
```

The output from the growaggr command provides first the size of the logical volume (here it is 12288 kilobytes, or 12 megabytes) and second the size of the aggregate. Both numbers are in units of blocks where each block has 1024 bytes.

To extend the size of the logical volume (in our example by 1 logical partition or 4 MB) use the extendlv command or SMIT extendlv:

```
# extendlv lfsuser3 1
```

If you issue the growaggr again, you will then see that the logical volume is now larger than the aggregate

```
# growaggr -aggregate lfsuser3 -noaction
aggregate lfsuser3 holds 16384 (1024B) blocks, 12288 assigned
```

5.3.7.3 How to Extend the Size of an Aggregate

The logical volume now has 16384 blocks, the aggregate has 12288 blocks. Since the logical volume has enough space left to increase the aggregate we can now actually grow the aggregate for example to 16000 blocks with the following command.

```
# growaggr -aggregate lfsuser3 -aggrsize 16000
```

Note that if you do not specify the -aggrsize option, you will grow the aggregate to the end of the logical volume.

To verify the action on the logical volume and on the aggregate issue again the following command:

```
# growaggr -aggregate lfsuser3 -noaction
aggregate lfsuser3 holds 16384 (1024B) blocks, 16000 assigned
```

The aggregate has still not reached the size of the logical volume, which allows us to increase the aggregate by another 384 blocks at a later time again. Let us check now how much space is left in the aggregate for filesets to grow:

```
# fts aggrinfo -s /.:/hosts/sys3 -aggregate lfsuser3
LFS aggregate lfsuser3 (/dev/lfsuser3): 15811 K free out of total 15992
```

First note, that the output shows only 15992 instead of 16000 blocks for the total size of the aggregate. Some blocks are used for header and administration. There are 15811 blocks left for filesets to grow.

5.3.8 Quotas on Filesets

Filesets do not have an initial size in DFS. Several DCE LFS filesets can compete for the space provided by the aggregate. The size of individual filesets can be obtained with the `fts lsheader` command.

```
# fts lsheader -s /.:/hosts/sys3 -aggregate lfspeter
Total filesets on server /.:/hosts/sys3 aggregate lfspeter (id 1): 4
user3.peter              0,,220 RW      8 K alloc     17 K quota On-line
user3.peter.backup       0,,222 BK     17 K alloc     17 K quota On-line
```

This example shows a fileset user3.peter with 17 kilobytes and the backup version user3.peter.backup with 17 kilobytes on the aggregate lfspeter.

A fileset is the unit that DFS administrator works with in order to move, replicate and do backups of user data. The DFS administrator can set a limit for the size of a fileset. When this limit is reached, the administrator can be notified to increase the limit (or the user can remove some of his unwanted files). This limit is called a *quota*.

An administrator can allocate more space to a particular user than is actually available. Whenever a user's fileset reaches the quota value, the administrator may increase the fileset quota or the user might have to delete some files. If the aggregate containing the fileset is full, the administrator could increase the size of the aggregate or the administrator could move the fileset to an aggregate that contains more room and then increase the quota for the fileset.

Note on quota

The sum of all quotas from several filesets within one aggregate can exceed the size of the aggregate.

Non-LFS filesets have a size which is equivalent to the logical volume. The quota of a fileset can be queried with the `fts lsquota` command. See the following examples:

```
# fts lsquota -path /:/user3/niki
Fileset Name          Quota    Used  % Used   Aggregate
user3.niki             5000      17     0%     4% = 188/4088 (LFS)
#
#
# fts lsquota -fileset user3.niki
Fileset Name          Quota    Used  % Used   Aggregate
user3.niki             5000      17     0%     4% = 188/4088 (LFS)
#
# cd /:/user3/niki
# fts lsquota
Fileset Name          Quota    Used  % Used   Aggregate
user3.niki             5000      17     0%     4% = 188/4088 (LFS)
#
```

The example demonstrates that you can use different options to specify the fileset. The output shows:

- Name of the fileset (user3.niki)

- the Quota in kilobytes (5000)
- Kilobytes currently in use (17)
- Percentage of used space of your quota range (0%)
- Percentage of real used space in your aggregate (4%)
- Kilobytes currently in use of aggregate (188)
- Total number of kilobytes on aggregate (4088)
- Type of fileset (DCE LFS)

For setting the quota on DCE LFS filesets the fts setquota command is provided. See the following example, which enlarges the quota on fileset user3.peter to 5500 kiloblocks.

```
# fts setquota -fileset user3.peter -size 5500
```

5.3.9 Locking and Unlocking of Filesets

The Fileset Location Server automatically locks entries in the FLDB for filesets before certain fts operations are executed on them and unlocks them thereafter. For the duration of the lock the fileset is prevented from being moved, copied, deleted or otherwise manipulated thus avoiding inconsistencies on the fileset. DFS also provides a manual method of locking and unlocking filesets. Normally, these will not be used.

The following commands determine if a lock is established and manually lock and unlock filesets in the FLDB

```
# fts lsfldb -fileset user3.peter
# fts lock -fileset user3.peter
# fts unlock -fileset user3.peter
```

5.3.10 Server Entries in the FLDB

Every DFS File Server must have an entry in the FLDB. The File Server entry in the FLDB stores information about the Fileset Server Machine such as its network address, the number of filesets exported and its principal name. You can see a single server entry by providing the server option or all of the server entries with the following commands.

```
# fts lsserverentry -server /.:/hosts/sys3
sys3.dce-dfs.itsc.austin.ibm.com (2:0.0.192.100.100.3)
FLDB quota: 0;  uses: 11;  principal='hosts/sys3';  owner=<nil>
#
#
# fts lsserverentry -all
sys1.dce-dfs.itsc.austin.ibm.com (2:0.0.192.100.100.1)
FLDB quota: 0;  uses: 12;  principal='hosts/sys1';  owner=<nil>

sys4.dce-dfs.itsc.austin.ibm.com (2:0.0.9.3.1.25)
FLDB quota: 0;  uses: 0;  principal='hosts/sys4';  owner=<nil>

sys2.dce-dfs.itsc.austin.ibm.com (2:0.0.192.100.100.2)
FLDB quota: 0;  uses: 4;  principal='hosts/sys2';  owner=private2

sys3.dce-dfs.itsc.austin.ibm.com (2:0.0.192.100.100.3)
FLDB quota: 0;  uses: 11;  principal='hosts/sys3';  owner=<nil>
```

Note the quota on the number of filesets which can be exported from a File Server. It is set to zero by default, which means that the quota is disabled. A number greater than zero will restrict a file server from exporting more than this number of filesets.

In general, the administrator does not have to create or delete entries for File Servers in the FLDB. The Server entry is created when you configure a DFS File Server and it is deleted when you unconfigure it. However there are commands to perform these actions. Examples for deletion and creation of server entries are listed here for a DFS File Server named /.:/hosts/sys4 and /.:/hosts/sys5.

```
# fts delserverentry -server /.:/hosts/sys4
# fts crserverentry -server /.:/hosts/sys5 -principal hosts/sys5/dfs-server -quota 20
```

It is more likely that the system administrator wants to change the quota or add another network address to the server entry. Here is an example of how to change the quota and how to add another network address.

```
# fts edserverentry -server /.:/hosts/sys3 -quota 20
```

Here is an example of adding another address for the host sys3. Note that up to four addresses can be established for a file server.

```
# fts edserverentry -server /.:/hosts/sys3 -addaddr 9.3.1.59
```

5.3.11 Replicating Filesets

Note

DCE/6000 Version 1.2 for use with AIX 3.2.5 does not contain the replication function. This will be supported in a future release.

Replicating a fileset means to copy a read-write fileset from a File Server machine and placing a read-only copy called replica on another File Server Machine. Both systems, where the master and the replica reside, must be configured as DFS File Server Systems and run the File Exporter processes. DFS provides replication of DCE LFS filesets (replication of a non-LFS fileset is not supported). The three main reasons for replicating filesets are:

1. Reduce load on a particular File Server System
2. Increase availability of a fileset
3. Decrease the load on slow network connections

The replica of a fileset is always read-only. This implies that when the master read-write fileset changes then the replicas need to be updated. Since updating a fileset via the network can cause heavy traffic and also be very I/O demanding you don't want to replicate filesets that require frequent changes. This would mean that the system has to constantly update it's own replicated sites. You would prefer to chose filesets which are mostly read-only type filesets and if updated then only for a short period of time (such as once a week).

Replication of a fileset requires that all filesets which are located higher in the hierarchy of the DFS filespace must also be replicated, including the root.dfs.

There are two types of replication:

1. Release Replication (manual)

2. Scheduled Replication (automated)

Release replication requires that the system administrator determines the time when to update the replica because of changes in the master fileset and the system administrator has to manually activate the procedure. With scheduled replication, the system administrator sets a schedule and the update procedure is started automatically.

Before any of the user filesets can be replicated you need to replicate the root.dfs and rearrange the access method for the read-write root.dfs.

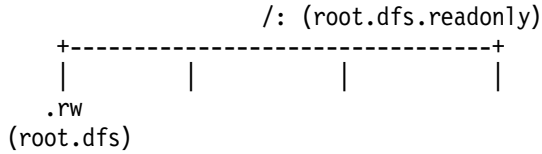
```
# fts crmount -dir /.:/.rw -fileset root.dfs -rw
```

This command mounts the root.dfs with a read-write mount point under its own path name so that all files have two path names now. One is /:/dir/dir/file and the other is /:/.rw./dir/dir/file. So far these are equivalent, except that /:/.rw is an explicit read-write mount point allowing read-write access to all files.

You must now replicate the root.dfs. The following commands will create a dfs.root.readonly fileset on the same machine where root.dfs resides

```
# fts setrepinfo -fileset root.dfs -release
# fts addsite -fileset root.dfs -server /.:/hosts/sys1 -aggregate root.lfs
# fts release -fileset root.dfs
```

This will result in the DFS client accessing the root.dfs.readonly file system whenever it traverses the /.../cell_name/fs pathname. The structure of your filespace looks like the following:



For more information about the concept of replication see 3.6, “DFS Replication - What It Is and How It Works” on page 67.

The following steps outline the procedure to create a replica of fileset user3.ron on system sys3 and place the replica on system sys1. Use Figure 44 on page 110 to understand the lab layout used with this example.

- The first step required is to define the type of replication on system sys3. The two options you have are release or scheduled replication. Here is an example with release replication.

  ```
  # fts setrepinfo -fileset user3.ron -release
  ```

  This command sets the type to release replication of fileset user3.ron. The `fts setrepinfo` command has plenty of options to define replication parameters such as times how long data are valid in the cache if the DFS Cache Manager uses data from read-only replication sites.

  Note

  Use the default parameters on the `fts setrepinfo` command unless you know exactly why you want to change any of these times.

- In the next step, you must define the site where the replica will be stored. Use the fts lsheader command to get the current size of the replica and then check if the target site has enough space to hold the replica with the fts aggrinfo command.

  ```
  # fts lsheader -server /.:/hosts/sys3 -aggregate lfs.sys3
  # fts aggrinfo -server /.:/hosts/sys1 -aggregate lfs.sys1
  ```

  If there is enough space available you can add the replication site

  ```
  # fts addsite -fileset user3.ron -server /.:/hosts/sys1 -aggregate lfs.sys1
  ```

  If there are other sites beside system sys1 where you want to replicate the fileset user3.ron, then you must repeat the fts addsite command once for each target site. The information is stored in the FLDB, and we could see the status of the fileset in the FLDB changing from invalid to valid.

- The last step shows how to actually create the read-only replicas at the sites that have been defined as replication sites. For release replication, a replica must always be created and existing at the master site first, before replication sites are able to copy the replica at the master site.

  ```
  # fts release -fileset user3.ron
  ```

  This command assumes the release replication type as defined above. It will create the read-only version at the master site. The replication server systems will then copy the read-only version to their machines and place the copies into the aggregates defined with the fts addsite command.

  The fts release command is not required with scheduled replication because creating (or updating) the read-only version is done automatically based on the schedule defined with setrepinfo.

At any given time and independent of release or scheduled replication type, you can enter the fts update command to force an update to the read-only version, such as in the following example:

```
# fts update -fileset user3.ron -server /.:/hosts/sys3
```

There are two commands available to monitor the status of replication server systems. The bos status command and the fts statrepserver command. Replication server systems are performing similar functions as file server systems. Therefore you must ensure that the ftserver process as well as the repserver process are running normal on the replication server system. This can be verified with the commands:

```
# bos status -server /.:/hosts/sys1 -p repserver
# bos status -server /.:/hosts/sys1 -p ftserver
```

In addition to the previous commands, you may issue the fts statrepserver command with the -long option to determine if the replication server is functioning.

```
# fts statrepserver -server /.:/hosts/sys3 -long
```

Or you may use the fts lsreplicas command to verify that all replicas of a fileset are functional:

```
# fts lsreplicas -fileset user3.ron -all
```

5.3.12 Cloning (Making a Backup) Filesets

A clone of a fileset is a snapshot of what a particular fileset looks like at a point in time. The result of a clone operation is a .backup file. The terms ″clone″ and ″backup filesets″ mean the same thing. Cloning of filesets is only supported with DCE LFS filesets. It is useful for the following reasons:

1. The backup version can be mounted in the DFS filespace and the user can retrieve files which were in the fileset at the time of backup. Therefore a user has a method to retrieve files that he may have accidentally deleted from his read/write fileset.
2. The backup version of a fileset can be used to provide input for the DFS Backup System Machine. Since the DFS Backup System Machine makes the fileset inaccessible for the duration of the dump, using the backup fileset reduces the time that the read-write version is inaccessible. (It takes less time to clone a fileset than it takes to back the file system up to tape using the DFS Backup System. Therefore the user′s down time is minimized.)

5.3.12.1 How to Clone (Make a Backup Version) of a Fileset

DFS provides the `fts clone` and the `fts clonesys` commands to create backup versions of LFS filesets. With `fts clone` you can backup a single readWrite LFS fileset. The `fts clonesys` command allows you to back up multiple readWrite LFS filesets at one time. We will now demonstrate how to create backup filesets. When new DCE LFS filesets are created and these filesets are entered into the FLDB there will be three entries visible with the `fts lsfldb` command.

```
# fts lsfldb
       .
       .

user3.andrea
        readWrite   ID 0,,244  valid
        readOnly    ID 0,,245  invalid
        backup      ID 0,,246  invalid
number of sites: 1
   server           flags     aggr   siteAge principal      owner
192.100.100.3       RW       lfsniki 0:00:00 hosts/sys3     <nil>
       .
       .
```

As you can see, there are three entries automatically generated in the database: the readWrite, the readOnly, and the backup version. The readOnly and the backup version show a status of invalid.

With the `fts clone` command we can create a backup version of the fileset user3.andrea:

```
# fts clone -fileset user3.andrea
Created backup fileset for user3.andrea
```

If you look again into the File Location Database, then you will see that the status of the backup version of fileset user3.andrea has now changed to valid.

```
# fts lsfldb
       .
       .

user3.andrea
        readWrite   ID 0,,244  valid
```

```
        readOnly    ID 0,,245  invalid
        backup      ID 0,,246  valid
number of sites: 1
   server           flags     aggr   siteAge principal      owner
192.100.100.3       RW       lfsniki 0:00:00 hosts/sys3     <nil>
       .
       .
```

The fileset which we have created has the name of the readWrite version with an extension of .backup. You can also use the `fts lsheader` command to verify that it has been created.

5.3.12.2 How to Read Files From Backup Filesets

To actually be able to access the files in this new fileset we must create a mount point for the backup LFS fileset.

```
# fts crmount -dir /:/user3/andrea/.Oldfiles -fileset user3.andrea.backup
```

You can read all files in the backup version, but you can not write to backup filesets. This allows you to restore files by copying them from the .Oldfiles fileset into your original readWrite fileset.

5.3.12.3 How to Clone (Backup) Multiple Filesets

The DFS system administrator usually does not want to perform clones of each and every single fileset. Therefore DFS provides the `fts clonesys` command which allows the administrator to clone several filesets in one step. See the following examples:

```
# fts clonesys -prefix user
# fts clonesys -server /.:/hosts/sys3
# fts clonesys -aggregate lfsusers  -prefix user
```

The first example performs a backup on all DCE LFS filesets with the prefix name of user such as user.peter or user.andrea. The second example limits the clonesys command to all filesets on server /.:/hosts/sys3 and the third example will only apply to those filesets with the aggregate name lfsusers that have the prefix name of user. Remember that backup versions can only be created from LFS filesets. You will get a warning message when trying to create a backup version of a non-LFS fileset.

5.3.12.4 How to Automate the Creation of Backup Filesets

For the system administrator it is probably useful to maintain backups on a fixed schedule and to create these backups automatically without administrator intervention. The method to do this is to create a cron entry in the BosConfig file. The bosserver process will then start the backup procedure automatically. For example, issue the following command:

```
# bos create /.:/hosts/sys3 backupusers cron '/opt/dcelocal/bin/fts \
clonesys -prefix users -localauth' 5:30
```

This command will run every day at 5:30 a.m. and create backups of all LFS filesets which have a name that starts with users.

5.3.13 Synchronize FLDB and Fileset Header

A DFS client that requests access to a DFS file first contacts the FLDB. The FLDB then provides a list of addresses of file servers that have the real fileset. The DFS client then contacts one of the file server for the file. This procedure implies that the FLDB has to have up-to-date information of the fileset itself. The entries in the FLDB and the fileset headers on the File Server systems must be accurate and synchronized.

There are two commands to synchronize the FLDB and the fileset header, `fts syncfldb` and `fts syncserv`. The `fts syncfldb` command examines the fileset headers and updates the FLDB if necessary. The `fts syncserv` command works in the other direction by examining the FLDB and updating the File Server header information if necessary. Examples for these commands are:

```
# fts syncfldb -server /.:/hosts/sys3 -aggregate lfsusers3
# fts syncserv -server /.:/hosts/sys3
```

For more information on when to use these commands, please see 8.3.6, “When Do You Need to Synchronize the FLDB and Fileset Headers?” on page 208.

5.3.13.1 Synchronization of non-LFS Filesets

Non-LFS filesets do not have fileset headers. If you use the `fts syncfldb` command on a server that has JFS filesets and the server and the FLDB are not in sync, then entries will be created for the JFS filesets in the FLDB. The fileset names given to these filesets in the FLDB will be of the form ″SYNCFLDB-ADDED-number″ (where ″number″ is a unique number to identify the fileset) since the fileset name cannot be obtained. (This is due to the fact that JFS filesets do not have fileset headers.)

The following is an example of resynchonizing the FLDB and making the non-LFS filesets usable again.

```
# fts syncfldb -server /.:/hosts/sys1
-- done processing entry 1 of total 2 --
-- done processing entry 2 of total 2 --
-- done processing entry 1 of total 1 --
-- done processing entry 1 of total 1 --
-- done processing entry 1 of total 1 --
Could not fetch FLDB entry (id=0,,151, type=0)
Error: FLDB: no such entry (dfs / vls)
        readWrite   ID 0,,151  valid
        number of sites: 1
           server           flags     aggr   siteAge principal      owner

sys1               RW       /ronjfs 0:00:00 hosts/sys1     <nil>

-- done processing entry 1 of total 1 --
FLDB synchronized with server /.:/hosts/sys1

# fts lsfldb -server /.:/hosts/sys1
SYNCFLDB-GENERATED-0-151
        readWrite   ID 0,,151  valid
        number of sites: 1
           server           flags     aggr   siteAge principal      owner

sys1               RW       /ronjfs 0:00:00 hosts/sys1     <nil>
# fts lsmount /:/ronjfs
'/:/ronjfs' is a mount point for fileset '#ronjfs'
```

```
# cd /:/ronjfs
/bin/ksh: /:/ronjfs: 0403-036 The specified directory is not valid.
```

Note that you cannot change to the /:/ronjfs directory because DFS cannot find the ronjfs fileset in the FLDB. The fileset known by the FLDB is named SYNCFLDB-GENERATED-0-151 and it must be renamed back to ronjfs so that DFS clients can then access it.

```
# fts rename SYNCFLDB-GENERATED-0-151 ronjfs
Renamed fileset SYNCFLDB-GENERATED-0-151 to ronjfs
# cd /:/ronjfs
```

5.4 DFS Monitoring & Controlling

There are two basic methods of monitoring the activity in a DFS environment:

- One is the scout program, which can be used to periodically monitor the load and disk usage on File Server systems.
- The other is the bosserver process, which monitors and restarts all DFS server administrative processes. The bos command suite is available for the system administrator to maintain the server processes.

5.4.1 Scout Program

The scout program allows system administrators to periodically collect information from File Server Systems about DFS data requests and displays this information on the local screen (ASCII or hft). In addition, the administrator can set thresholds on the statistics that scout displays. If a threshold is reached, the program highlights the displayed data to warn the administrator. Scout allows information to be displayed from several host systems within the local cell and/or from other cells.

The scout program can be started from any DFS client or server system without any configuration steps. If you are using X-Windows, ensure that the TERM variable is set (TERM=aixterm works) and that your window is sized properly. For good results, widen your window as much as possible and set the length of the window to a size which can hold the list of all systems which you intend to display. Then call the scout program. It must run in the foreground of a window and this window cannot be used to enter any other commands as long as Scout runs. Activate the Scout Program as follows:

```
# scout -server sys1 sys2 sys3 sys4 -basename /.:/hosts -frequency 10
```

In this example, the parameter -basename /.:/hosts is a prefix for the name of all DFS File Server systems provided with the parameter server. This command will monitor the File Exporters on File Server machines named /.:/hosts/sys1 through /.:/hosts/sys4 by taking probes every 10 seconds. The output from the Scout program has the following format:

```
                                Scout for /.:/hosts

Conn     Fetch      Store    Ws      Server     Disk attn: > 95% used
----    -------    -------  ----  ------------  ----------
   7        415       1539     5      sys1      user1_lv1:40407:*ort/sys1-jfs.1:0:
                                                *ys1-jfs.2:3932:*3:*er1_lv2:36551:*
   5         10          2     5      sys2      lfs.sys2:8022:*xport/proj2:3920:
                                                *2.2:7987:*
   6         28        708     5      sys3      lfspeter:3900:*.user3:10939:
                                                *ort/janice:7904:
                                                *tom:3936:*iki:3892:*
   0          0          0     1      sys4      /export/proj4:3936:*1:16167:*
```

The output of Scout can be interpreted as:

- Conn: The number of open connections with clients
- Fetch: The number of blocks of 64 kilobytes which client machines have fetched from the File Exporter since the file exporter was restarted.
- Store: The number of blocks of 64 kilobytes which client machines have sent to the File Exporter for storage since the file exporter was. restarted.
- Ws: The number of active client machines in the last 15 minutes.
- Server: The File Server Machine name that these statistics apply to.
- Disk attn: The disk usage. It shows the number of available kilobytes on each aggregate on the File Server machine. Aggregate names are abbreviated so that only distinctive letters at the end are visible. This makes the names hard to read in some cases.

To stop the Scout program, just enter **<CTRL_C>**.

A nice feature on the Scout program is the option to set thresholds. A threshold can be applied to all columns except the File Server machine name. A default threshold of 95% is automatically set for the maximum percentage of data in an aggregate. Each aggregate which exceeds this threshold will be highlighted. See the following example to set the threshold to a 90% value:

```
# scout -server sys1 sys2 sys3 sys4 -basename /.:/hosts -attention disk 90%
```

You must enter the % (percent sign) with the disk threshold. If the % sign is absent, Scout interprets the number as a threshold for the number of kilobyte blocks left. If there are less than this number of disk kilobyte blocks free in any aggregate you get a warning. For example

```
# scout -server sys1 sys2 sys3 sys4 -basename /.:/hosts -attention disk 5000
```

This means that you will get a warning if you have less than 5000 blocks free in the aggregates on sys1, sys2, sys3 or sys4.

Note

See the scout command in InfoExplorer for other parameters that can be highlighted

5.4.2 BOS Server program

The Basic OverSeer (BOS) Server, or bosserver process runs on every DFS server system. There is no need to run the bosserver process on DFS client systems. Its primary function is to supervise the majority of other DFS server processes and to restart these processes when necessary. The purpose is to reduce the need for an administrator to monitor these processes. The bosserver process itself runs without an operator once DFS is started. (This bosserver process is started automatically when you configure the various DFS servers.)

5.4.2.1 BOS Configuration File (BosConfig)

The processes which are to be supervised and monitored by bosserver must be entered into a configuration file (/opt/dcelocal/var/dfs/BosConfig) on the system where the bosserver process runs. These processes can either be DFS processes or user defined processes. This file is updated with commands from the bos command suite. Let us look at an example of the BosConfig file.

```
# pg /var/dce/dfs/BosConfig
restarttime 11 0 4 0 0
checkbintime 3 0 5 0 0
bnode simple upclient.scm 1
parm /opt/dcelocal/bin/upclient -server /.:/hosts/sys1   -path /opt/dcelocal/var
/dfs/admin.bos  /opt/dcelocal/var/dfs/admin.bak  /opt/dcelocal/var/dfs/admin.fl
 /opt/dcelocal/var/dfs/admin.ft
end
bnode simple ftserver 1
parm /opt/dcelocal/bin/ftserver
end
```

The restarttime line in this file specifies when the processes in this file (marked as running) should be restarted. Please see 5.4.2.4, “How to Use bos to Stop, Remove and Restart Processes” on page 138 for more details.

The checkbintime line in this file specifies a restart time for running processes that have new binaries stored in /opt/dcelocal/bin.

The sequence of lines starting with bnode and ending with end define a process entry in the BosConfig file. Our example holds two process entries: the upclient.scm and the ftserver process. Each process entry has the following fields in the BosConfig file:

1. The type of the process: simple or cron. A simple process is a process that is started once and runs continuously. A cron process is a process that runs at specified times.
2. The name of the process: can be any given name, but there are recommended names for DFS server processes.
3. The status of the process: 1 or 0, which stands for running or not-running. When a process is marked as running, bos starts it at reboot and also automatically whenever bos detects it to be not running.
4. The parameters of the process used by the bosserver to run the process.

The bosserver process keeps a copy of the BosConfig file in memory. Some bos commands only affect the in-memory copy of the BosConfig file. In other cases they affect only the BosConfig file. It is therefore possible that the real status of a process does not match the entries in the BosConfig file. For example, a process may be running even if the status flag shows *not-running*.

Warning

Do not edit the BosConfig file with an editor or with any other shell command. Use only the bos command suite to make changes to this file.

5.4.2.2 How to Use bos to Create and Start bosserver Processes

You can use the bos create command to create a process and also to immediately change its state to running. This command creates a new entry in the BosConfig file.

Let us look at an example. Assume we have created a shell script called ps_to_log which writes a table of all running processes to /opt/dcelocal/var/dfs/adm/PsLog.

```
#cat ps_to_log
#!/bin/ksh
ps -eaf > /opt/dcelocal/var/dfs/adm/PsLog 2>&1
```

If we want to have this shell script running every Monday at 2:30 pm, then we would have to enter the following command:

```
# bos create -s /.:/hosts/sys3 -p my_own_creation -type cron \
> -cmd '/tmp/ps_to_log' 'mon 2:30 pm'
```

This would create an entry in the BosConfig file such as:

```
bnode cron my_own_creation 1
parm /tmp/ps_to_log
parm mon 2:30 pm
end
```

and the bosserver process that is located on host sys3 would run this process every Monday at 2:30 pm.

Note use of #!/bin/ksh

It is important to have that first line saying #!/bin/ksh in the shell script. This is a directive to the bosserver to use the ksh.

5.4.2.3 How to Use bos to Check the Status of bosserver Processes

The command bos status checks the status of BosConfig defined processes on DFS server systems and displays the output on the screen. See the following example, which displays the status of all processes controlled by bosserver on system sys1:

```
# bos status -s /.:/hosts/sys1 -long
Instance ftserver, (type is simple) currently running normally.
    Process last started at Mon Apr 25 11:00:00 1994 (1 proc starts)
    Parameter 1 is '/opt/dcelocal/bin/ftserver'

Instance upserver, (type is simple) temporarily disabled,
 stopped for too many errors, currently shutdown.
    Process last started at Mon Apr 25 11:02:48 1994 (13 proc starts)
    Last exit at Mon Apr 25 11:02:58 1994
    Last error exit at Mon Apr 25 11:02:58 1994, by exiting with code 2
    Parameter 1 is '/opt/dcelocal/bin/upserver -path
    /opt/dcelocal/var/dfs/admin.bos
```

```
  /opt/dcelocal/var/dfs/admin.bak  /opt/dcelocal/var/dfs/admin.fl
/opt/dcelocal/var/dfs/admin.ft'

Instance bakserver, (type is simple) currently running normally.
    Process last started at Mon Apr 25 11:00:01 1994 (1 proc starts)
    Parameter 1 is '/opt/dcelocal/bin/bakserver'
```

You may also use the bos status command to display output of just a single process by naming this process with the option -process <name_of_process> such as:

```
# bos status -s /.:/hosts/sys3 -process my_own_creation -long
```

5.4.2.4 How to Use bos to Stop, Remove and Restart Processes

There are several commands to stop, remove and restart processes from the bosserver. Since some of the bos control commands take action only on the flags in memory and not in the BosConfig file, it is recommended to use the commands only in the following combinations.

To stop and remove a process from the BosConfig file, use (this will change the run flag in both memory and the BosConfig file)

```
# bos stop   -s /.:/hosts/sys3 -p name_of_process
# bos delete -s /.:/hosts/sys3 -p name_of_process
```

To stop and restart a process temporarily (this will change the run flag only in memory)

```
# bos shutdown  -s /.:/hosts/sys3 -p name_of_process
# bos startup   -s /.:/hosts/sys3 -p name_of_process
```

To stop and restart a process (this will change the run flag in the BosConfig file and memory)

```
# bos stop  -s /.:/hosts/sys3 -p name_of_process
# bos start -s /.:/hosts/sys3 -p name_of_process
```

The -wait option can be used on bos stop and bos shutdown commands and cause the command shell prompt to remain absent until the process has stopped.

In case of having installed new binaries it may be necessary to just stop and restart all server processes including the bosserver itself. This can be achieved with the following command:

```
# bos restart -s /.:/hosts/sys3 -bosserver
```

You can check or set default restart times with bos getrestart or bos setstart commands. See the following examples:

```
# bos getrestart -s /.:/hosts/sys4
Server /.:/hosts/sys3 restarts at sun 4:00 am.
Server /.:/hosts/sys3 restarts for new binaries at 5:00 am.

# bos setrestart -s /.:/hosts/sys4 -general "sun 4:01 am"
# bos setrestart -s /.:/hosts/sys4 -newbinary "6:03 am"
# bos getrestart -s /.:/hosts/sys4
Server /.:/hosts/sys3 restarts at sun 4:01 am.
Server /.:/hosts/sys3 restarts for new binaries at 6:03 am.
```

5.4.2.5 How to Reboot a DFS Server Machine

Rebooting a server machine is not difficult, but think twice before you perform this step. There could be dependencies of other users in your cell that may experience loss of service for the time when your machine reboots. For example, DCE server functions such as security services, CDS services, FLDB services and others can turn the complete cell into a non-operational state.

In case you have to reboot follow these steps:

```
# su root
# shutdown -rF
```

The above command sequence shuts down all DFS server processes on the sys3 machine. After they are shutdown, a machine shutdown and reboot of sys3 is performed.

5.4.2.6 Log Files

Most of the server processes and also the bosserver process itself generate log files. The log files are located in /opt/dcelocal/var/dfs/adm directory. These log files are:

- BakLog: generated by the backup server process (bakserver)
- BosLog: generated by the bosserver process
- FlLog: generated by the Fileset Location Server (flserver)
- FtLog: generated by the Fileset Server (ftserver)
- RepLog: generated by the Replication Server process (repserver)
- UpLog: generated by the Update Server process (upserver)

These files can be viewed with standard AIX commands, but you may also use the `bos getlog` command such as:

```
# bos getlog -server /.:/hosts/sys3 -file BosLog
```

The main benefit of using the `bos getlog` command to review these files is that you can specify which server's log files you want to view without having to log into that machine.

Sending a "kill -30" signal to DFS server administrative processes result in icl.* files being created. These icl.* files may be useful in trouble shooting situations. These icl.* files will be helpful for IBM support personnel although the administrator may also be able to to discover information that may help him to resolve the problem. The DFS server administrative processes that this applies to are the DFS processes that have CDS entries as well as the dfsbind process.

5.5 Backup and Restore with DFS

There are several ways to maintain backups on filesets or aggregates with DFS. These are:

1. The DFS Backup System, a sophisticated method supported by a database to hold information about the dumps.
2. The DFS `fts dump/restore` commands which handle both filesets as well as complete aggregates.

3. The standard UNIX commands such as `tar`, `cpio`, `dd`, `backup/restore` and others. (Note that the DCE ACLs are not preserved during these operations.)

This section will deal mainly with the DFS Backup System but the other methods are described briefly. Since backup/restore is such an important concept please see Figure 48 for an overview of what methods may be used for backup/restore.

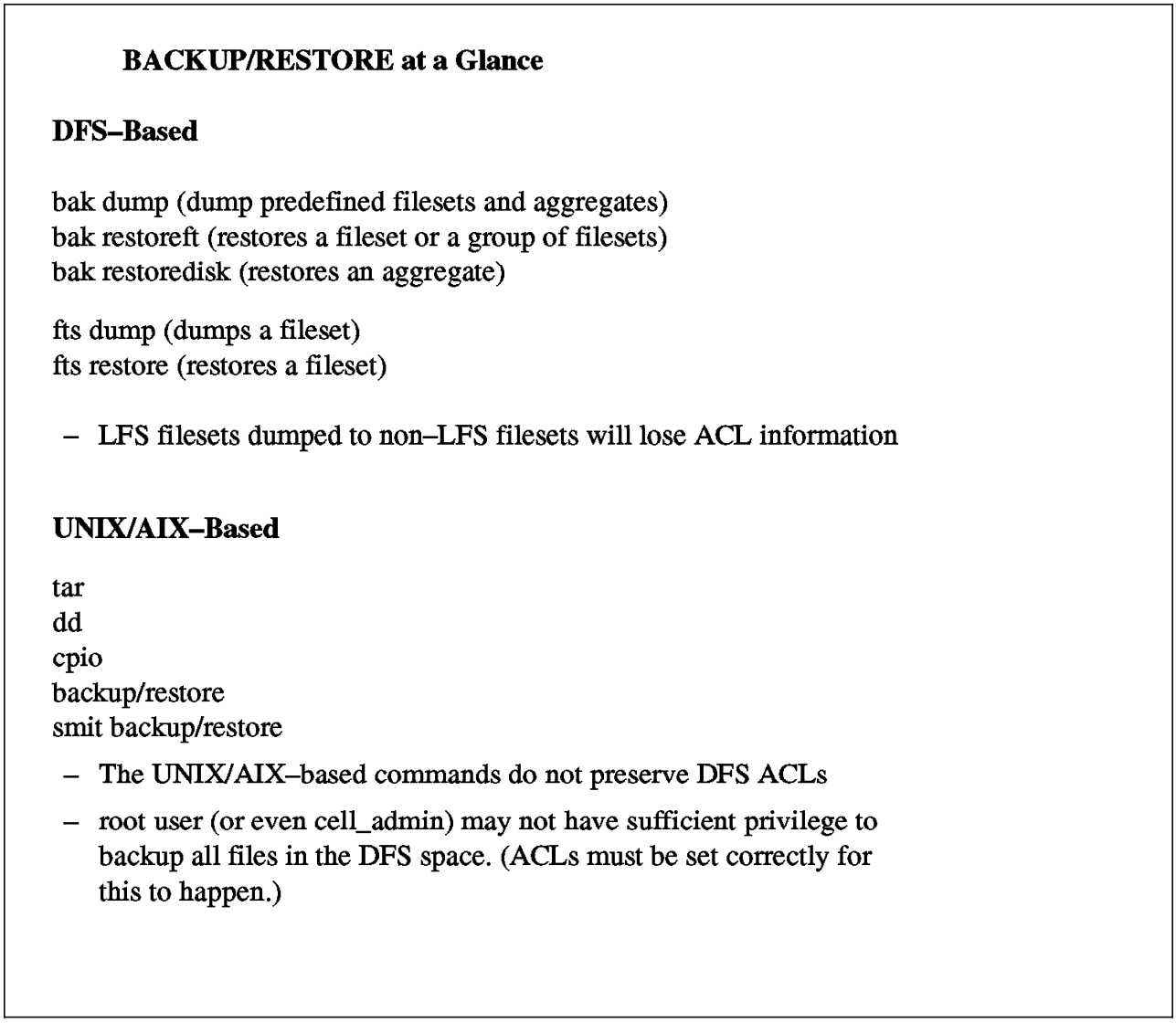


Figure 48. Different Methods and Considerations of Backup/Restore

If you are using the DFS Backup System you need to setup one of the machines as a DFS Backup Server. You also want to assign a system backup administrator. The default user to perform this job is the cell_admin who is already set up as a principal in the DCE cell. As an administrator for the backup/restore setup of your cell it is best to pick a system which has one or more tape units installed and configure this system as the Backup Database Machine and also as the Tape Coordinator System.

5.5.1 The DFS Backup System

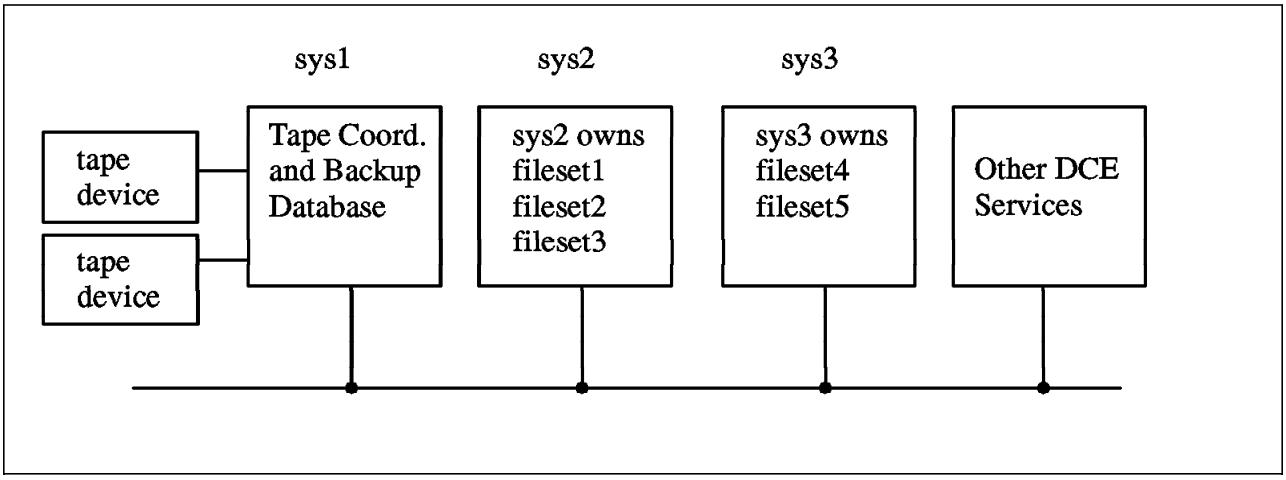


Figure 49. Overview of a DCE Cell Showing sys1 as the DFS Backup Machine

5.5.2 Configure the Backup Database Machine

To create a Backup Database machine you need to run the /usr/bin/mkdfs command with the following parameters:

 # mkdfs -s <dce_pathname of system control machine> dfs_bkdb

Make sure that you specify the SCM as the server parameter because the configuration process has to know from where the administrative lists are distributed. See 4.4.5.1, “Configuring a Backup Database machine” on page 103 for information on how to configure the Backup Database Machine with SMIT. This command will do the following:

1. It will create the basic structure the Backup Database in the /var/dce/dfs/backup directory. You may want to verify this by listing the contents of the directory.

   # ls /var/dce/dfs/backup
   bkdb.DBO bkdb.DBSYS1

   These files can not be viewed or edited with standard AIX commands.

2. It will add an entry to the BosConfig file, the bakserver program which is responsible for maintaining the Backup Database.

   ```
   bnode simple bakserver 1
   parm /opt/dcelocal/bin/bakserver
   end
   ```

   The bakserver program should also be started immediately by the bosserver program and if you issue the ps command you would see the process. This may also be verified with the following command

   # bos status -server <dce_pathname of name_of_server> -process bakserver

3. There is also an administration list created in directory /var/dce/dfs with the name admin.bak. If your system happens to be also the System Control Machine then you can maintain this file but if another system is acting as SCM you need to maintain admin.bak on the SCM, since it will be copied to all machines via the upserver/upclient processes and changes would be overwritten. List the contents of this administrative list:

```
# bos lsadmin -server <dce_pathname of name_of_server> -adminlist admin.bak
```

These are the users, who have privileges to control the Backup Database and to issue bak commands. Most likely you as the backup system administrator belong to the group subsys/dce/dfs-admin which by default is listed in the admin.bak file (and all the other admin groups). If you want some principals to only have access in a smaller domain, create a new group, put the user in that group and then add the group to the appropriate admin lists in that domain.

Recommendation

If the mkdfs dfs_bkdb command fails, run the rmdfs dfs_bkdb before you issue another mkdfs. This will clean up anything that has been generated with the first command.

5.5.2.1 Configure the Tape Coordinator

The Tape Coordinator is the system, where you will run the butc process (butc is an abbreviation for backup tape coordinator) for each physical tape attached. There is only a small configuration file which has to be set up for this. Create and edit the /var/dce/dfs/backup/TapeConfig file. An Example is shown here:

```
# vi /var/dce/dfs/backup/TapeConfig
2g   200k  /dev/rmt0   0
2g   200k  /dev/rmt1   1
2g   200k  /dev/rmt2   2
```

The first column is the tape size, the second is the EOF mark size, the third is the device name of the tape and the fourth is the Tape Coordinator ID. See also 3.5.2, “Tape Coordinator” on page 63 where we explain the fields in this list. To find the size of the tape and length of the gaps you can issue the fms command, but it takes approximately two hours for an 8mm tape with 2.3 GB capacity.

```
# fms -tape /dev/rmt0
wrote block: 142795
Finished data capacity testing - rewinding
wrote 10996 blocks, 109960filemarks
Finished filemark test
Tape capacity is 2339586048 bytes
File marks are 196362 bytes
```

It is recommended that the tape size returned by the fms command be reduced by 10 to 15 % and that the EOF mark size be increased by 10 to 15 % before being used in the TapeConfig file.

5.5.2.2 Populate the Backup Database

The first step is to add the Tape Coordinator entries to the Backup Database. There must be one Tape Coordinator for each entry you have put into the TapeConfig file. The Backup Database allows a maximum of 1024 Tape Coordinators. The command to add Tape Coordinator entries to the Backup Database is bak addhost. See an example which matches the entries we have placed into the TapeConfig file:

```
# bak addhost -tapehost /.:/hosts/sys3  -tcid 0
# bak addhost -tapehost /.:/hosts/sys3  -tcid 1
# bak addhost -tapehost /.:/hosts/sys3  -tcid 2
```

You must put one entry for each physical tape drive into the Backup Database. The entries in the Backup Database define:

- The machine name of the Tape Coordinator (-tapehost)
- The Tape Coordinator's TCID (-tcid)
- The UUID of the Tape Coordinator (generated automatically)

You may want to list the entries of Tape Coordinators in the Backup Database. This can be performed with the bak lshosts command.

```
# bak lshosts
Tape hosts:
     Host /.../dino/hosts/sys3, port offset 0
     Host /.../dino/hosts/sys3, port offset 1
     Host /.../dino/hosts/sys3, port offset 2
```

The next step involves setting up fileset families. Think about a family name that best describes the group of filesets which you want to place into the family. Use the bak addftfamily command to create a family. See the following example:

```
# bak addftfamily -fam user_data
```

Do not use periods in the family name because the dump level handling will use periods and indexes internally and it will get confused if you use them too.

Now we will add several filesets to the family which we just created. The command to perform this step is bak addftentry. The family to which you want to add filesets must already exist. Let us look at an example where we pick selected fileset to be included in the family called user_data

```
# bak addftentry -fam user_data -s /.:/hosts/sys3 -aggr  lfsusers \
-fileset "users.*"
# bak addftentry -fam user_data -s /.:/hosts/sys3 -aggr  /export/niki \
-fileset "users.*"
# bak addftentry -fam user_data -s ".*" -aggr  ".*" -fileset "*.backup*"
```

The first entry selects all filesets which start with the name users in the aggregate lfsusers on the system sys3. The second entry catches all filesets starting with the prefix users in the JFS aggregates named /export/niki on host sys3. The third entry matches all filesets with the extension .backup in the DFS administrative domain.

Note

It is recommended to clone filesets before using the DFS Backup System to actually dump the filesets to tape. This is because for the duration of the dump the fileset is inaccessible for other users. By using the clone for DFS Backup System, you reduce the time of inaccessibility to the time it takes to make the clone filesets (instead of the time it takes to backup the filesets to tape.).

The bak command suite also has parameters to list the information from the DFS Backup Database. Use bak lsftfamilies to list the families and also the filesets which you have placed in the families.

5.5.2.3 Define Backup Policies

The Backup System knows what it is supposed to backup and where to find the filesets. The next step involves to define policies how often backup are to be done and how long these dumps should be kept (expiration dates). The expiration date on a tape is used by the backup system to prevent you from mistakenly writing over the tape. If the date on the tape has not expired, a write to this tape will not be allowed. You may relabel this tape to remove the expiration date as follows:

```
# tctl -f /dev/rmt0 rewind
# tctl -f /dev/rmt0 weof 1
# tctl -f /dev/rmt0 rewind
# bak tapelabel
```

We define the backup policies with the `bak adddump` command. See the following example, which is also explained in 3.5.4, “Dump Levels” on page 65. The corresponding entries in the Backup Database for this backup are:

```
# bak adddump -l /sunday -expires in 1w
# bak adddump -l /sunday/mon -expires in 1d
# bak adddump -l /sunday/tue -expires in 1d
# bak adddump -l /sunday/wed -expires in 1d
# bak adddump -l /sunday/thu -expires in 1d
# bak adddump -l /sunday/fri -expires in 1d
# bak adddump -l /sunday/sat -expires in 1d
```

Verification of these entries into the Backup Database is possible with the `bak lsdumps` command.

```
# bak lsdumps
/sunday  expires in 1w
    /mon expires in 1d
    /tue expires in 1d
    /wed expires in 1d
    /thu expires in 1d
    /fri expires in 1d
    /sat expires in 1d
```

We have now defined to take a full dump every Sunday and to take incremental dumps every weekday. All incremental dumps reference the full dump from Sunday as their parent dump.

5.5.2.4 Start The Tape Coordinator

The Tape Coordinator process (butc) must be started for each physical tape attached to the system. The Tape Coordinator will use the TapeConfig file for configuration and you want to make sure that the TapeConfig holds a separate line for each tape unit. Start the Tape Coordinator with the following command

```
# butc -tcid <id_of_tape>
```

If you have several tapes in use and therefore several Tape Coordinators then you want to start each butc process in a separate X-Window session. The butc process must run in the foreground and the session is unavailable for any other commands. The butc process will instruct you what to do, for example to insert a new tape or remove a tape. The session must remain running as long as you perform any read or write operations with the tape. If you have finished you may stop the Tape Coordinator by entering an interrupt signal, such as <Ctrl_C>.

Warning

Do not exit from the butc program if there are still tape operations running.

5.5.2.5 Label Your Tapes

The tapes in use with the Backup System have labels which can be created by the system administrator. Labeling a tape involves writing a record to it that formally gives it a name. This is to help the backup administrator by keeping him from overwriting tapes that he intends to save. However the bak dump command accepts partially labeled or completely unlabeled tapes, and it will write the labels by itself. Tapes will always be labeled. This will be done either by the administrator using the bak labeltape command or by the butc process itself. If we use our example of dump hierarchy established in the previous step, we would need to label seven tapes, one for each day of the week. An example of the labeling procedure is shown here:

```
# bak labeltape -tape user_data.sunday.1
# bak labeltape -tape user_data.mon.1
# bak labeltape -tape user_data.tue.1
# bak labeltape -tape user_data.wed.1
# bak labeltape -tape user_data.thu.1
# bak labeltape -tape user_data.fri.1
# bak labeltape -tape user_data.sat.1
```

The full dump which is taken on Sunday may require more than one tape so that you have to label extra tapes as user_data.sunday.2 and so on. If you need to check on labels, there is also a command to read the labels. Use the bak readlabel command to read the labels from a tape.

5.5.2.6 Start The Dumps

The actual dumps are activated with the bak dump command. They are performed on families defined in the Backup Database. The commands to execute the full dump of family user_data which we had defined previously is:

```
# bak dump -family user_data -level /sunday -tcid 0
```

The Tape Coordinator (butc) process will request that you mount the tape labeled user_data.sunday.1 and all filesets defined in the family user_data will be dumped to tape. Then, a day later, on Monday, you would have to start the incremental dump with the following command:

```
# bak dump -family user_data -level /sunday/mon -tcid 0
```

A new tape with the tape label user_data.mon.1 will be requested by the Tape Coordinator for Monday's dump.

Note

The DFS Backup System is very particular about having the correct tape in the tape drive. It will compare the label on the tape and refuse all unacceptable tape labels such as those with an incorrect date or simply the wrong label.

For an automated procedure you want to put the bak dump commands into the crontab file of a user, who has the required privileges. This would be useful to automate the backup procedure, however, a system administrator still needs to respond to the messages from the butc process. Currently there is no complete automated procedure available with the butc process.

5.5.2.7 Restore DFS Dump Data

Data previously dumped with bak dump command can be restored either with bak restoreft or with bak restoredisk command. The bak restoreft command allows to restore a single fileset or a set of filesets which belong to the same family. The bak restoredisk command will restore the entire contents of an aggregate. A full restore includes data from the last full dump and all subsequent incremental dumps. It may therefore request several tapes to be inserted into the tape unit. If you want to check which tapes belong to a restore operation without actually performing the restore then it is helpful to issue the bak restoreft command with the -noaction option, such as

```
# bak restoreft -ser /.:/hosts/sys3 -aggregate lfs.user3 -f user3.peter -noaction
```

To actually restore a fileset and not overwrite the previous version you want to provide an extension, such as

```
# bak restoreft -ser /.:/hosts/sys3 -aggregate lfs.user3 -f user3.peter -extension .re
```

This command will restore the fileset user3.peter under a new name (called user3.peter.re) in the aggregate lfs.user3 on the server /.:/hosts/sys3. To be able to access it, we need to create a mount point.

```
# fts crmount -dir /:/user3/peter.re -fileset user3.peter.re
```

5.5.2.8 Scanning the Dump Tape

Issue the bak scantape command to scan the data which has been dumped with bak dump command.

```
# bak scantape -tcid 0
```

It will provide information (unfortunately in the butc window and not in the bak process window) about each fileset on the tape. A sample output is shown here:

```
-- fileset --
fileset name: root.dfs.backup
fileset ID 0
dumpSetName: dino_cell_backup.sun
dumpID 741987284
level 0
paretnID 0
endTime 0
clonedate Tue Jul 6 13:50:26 1993
```

This command also allows you to add the information extracted from the tape to the DFS Backup Database. This may be useful if the information from the tape somehow became corrupted in the Backup Database and you need to restore part of the database.

```
# bak scantape -dbadd -tcid 0
```

It is not possible to extract information about only a specific fileset on the tape. Only data about all filesets on the tape can be extracted. (Note that the bak ftinfo command may be used to display information about a fileset but this information comes from the backup database.)

5.5.2.9 View Information from the Backup Database

The following commands can be used to list information from the DFS Backup Database:

bak verifydb Checks the status of the Backup Database

bak lsftfamilies lists fileset families and entries in fileset families

bak lsdumps lists the entries in the dump hierarchy

bak dumpinfo displays information about specific backups

bak lshosts lists Tape Coordinators

bak ftinfo displays the dump history for a fileset from the backup database

5.5.2.10 Backup and Restore the Backup Database

Because it is very difficult to recreate information about dumps it is important to maintain a backup of the Backup Database itself. The bak savedb command is provided for copying the entire database to tape. You need to designate an extra tape for this procedure, which should be labeled bak_db_dump.1.

```
# bak labeltape -tape bak_db.dump.1
# bak savedb -tcid 0
```

If the Backup Database becomes corrupted (which can be tested with the bak verifydb command) you need to restore the database. Before you actually restore the database, you must delete the old database. The steps involved in this procedure are:

```
# bos stop -s <name_of_server> -p bakserver
# rm /var/dce/dfs/backup/bkdb.*
# bos start -s <name_of_server> -p bakserver
# bak addhost -tapehost <name_of_server> -tcid 0
```

Now restart the Tape Coordinator Machine by issuing in another window:

```
# butc -tcid 0
```

and then restore the Backup Database

```
# bak restoredb -tcid 0
```

If the restored database does not hold the most current information, you may use the bak scantape command with the -dbadd option, which allows you to extract information from the dump tape and add it to the Backup Database.

5.5.3 Using the fts dump and fts restore Commands

The commands fts dump and fts restore provide another method of backing up data. The fts dump command converts the content of a fileset into a byte stream which can be stored in a file or can be piped from standard output to other commands which accepts standard input. For example, to perform a dump of a fileset with the name user3.peter and store the dump in the /tmp directory enter the following command:

```
# fts dump -fileset user3.peter -time 0 -file /tmp/user3.peter.dump
```

The way to restore this fileset is by issuing the fts restore command, such as

```
# fts restore -ftname user3.peter -s /.:/hosts/sys3 -aggr lfspeter  \
-file /tmp/user3.peter.dump -overwrite
```

Although dumping a fileset works on all types of DFS filesets, you can restore only to the read-write version of the fileset. You can mix DCE LFS and non-LFS filesets while you are dumping and restoring but keep in mind that you lose ACL information when restoring an DCE LFS fileset to a non-LFS fileset.

The fts dump and fts restore commands provide an easy way of copying the read-only version of a fileset into the read-write version in case the read-write version is corrupted. Here is an example:

```
# fts dump -fileset user3.peter.readonly -time 0 | fts restore -ftname user3.peter \
-s /.:/hosts/sys3 -aggr lfspeter -overwrite
```

Note

The contents of the fileset are unavailable during the dump or restore operation. It is therefore better to start the dump only on the backup version of a fileset, which does not interrupt the access to the read-write version of the fileset.

5.5.4 Other Methods of Backup/Restore

There are other methods of backup and restore that are available with UNIX. In the following sections we note how these methods interact with the DCE DFS.

5.5.4.1 Use of the tar Command

The tar command can be used to archive files from the DFS filespace and to restore these files into the DFS filespace or non-DFS filespace.

```
# cd /:/user3/peter
# tar -cvf /dev/rmt0 file1 file2
# rm file1 file2
# tar -xvf /dev/rmt0
```

Note that the tar archive does not maintain DCE ACLs. For a robust discussion of ACLs, please see Chapter 6, “Security” on page 161. For example, if you have set some additional ACL definitions such as mask_obj or foreign_user then these ACL entries are gone after restoring the files. Look at the following figure to understand the impact of tar on ACLs. As you can see in our example, you not only lose the user:peter and the mask_obj parameters, but also the insert permission. The control permission on user_obj is always set. Otherwise the user could not change his own ACLs.

```
ACL for                                          ACL for
file1 in DFS Filespace                           file1 in DFS Filespace
before use of tar                                after use of tar restore
-----------------------                          -----------------------
mask_obj:  r-xcid
user_obj:  rwxcid                                user_obj:  rwxc--
user:peter:rwxcid   --> tar create/restore -->
group_obj: r-x---                                group_obj: r-x---
other_obj: r-----                                other_obj: r-----
```

Let us also look at an example when we create a tar archive from a file in the non-DFS filespace and restore this file into the DFS filespace.

```
File in non-DFS Filespace                       File in DFS Filespace
before use of tar                               after use of tar restore
-----------------------                          -----------------------
owner: rwx          --> tar create/restore -->   user_obj : rwxc--
group: r-x                                       group_obj: r-x---
other: r--                                       other_obj: r-----
```

When a file is backed up from a non-DFS filespace and restored into a DFS filespace, the UNIX permission bits are put into the ACL for that file. Other ACL entries are empty except the control permission on user_obj which always needs to be set so that the owner can change the ACL itself.

Warning

The use of the tar command sometimes requires the permission rights of user root as it is known from standard UNIX (for example, when restoring a read only directory). In DFS, the local root user has no extra privileges. Because root has no extra privileges for DFS files, it may therefore happen that common backup/restore programs in use today will not backup the desired files in DFS.

5.5.4.2 Use of the dd Command

The dd command can be used to write files from the DFS filespace and to restore these files into the DFS filespace or non-DFS filespace. Like the tar command, the dd command does not maintain DCE ACLs.

In this example, file2 will receive user_obj, group_obj and other_obj ACL entries but no entries for other ACL types. In addition, the user_obj, group_obj and other_obj ACL entries will *not* be the same as those in the ACL for the original file. The following output shows you the ACL settings on file1 and on file2 when the dd command is used:

```
# cd /:/user3/peter

# dd if=file1 of=file2
dd: 0+1 records in.
dd: 0+1 records out.
# acl_edit /:/user3/peter/file1 -l

# SEC_ACL for /:/user3/peter/file1:
# Default cell = /.../dino
mask_obj:rwxcid
user_obj:rw-c--
user:peter:rwxcid
group_obj:r-x---
other_obj:r-----

# acl_edit /:/user3/peter/file2 -l

# SEC_ACL for /:/user3/peter/file2:
# Default cell = /.../dino
user_obj:rw-c--
group_obj:r-----
other_obj:r-----
#
```

The settings of the UNIX permission bits is influenced by the umask, the initial object definition in the parent directory, and the internal function of the command. See 6.3.3, “AIX Group Mode Bits and DFS ACL Interaction Caused by mask_obj” on page 181 for a discussion of these parameters.

5.5.4.3 Use of cpio Command

The cpio command can be used to copy files from the DFS filespace and restore these files into the DFS filespace or non-DFS filespace. DCE ACLs are not maintained.

```
# cd /:/user3/peter
# find . -print | cpio -ov > /dev/rfd0
# cd /:/user3/tom
# cpio -idmv < /dev/rfd0
```

5.5.4.4 Use of backup and restore Command

This example shows a method to backup a single directory

```
# cd /:/user3/peter
# find . -print | backup -if /dev/rmt0
# rm -r /:/user3/peter/*
# restore -xvf /dev/rmt0
```

Also in this example you will lose the DCE ACLs. You may use the backup/restore commands on a complete file system in backup by I-node fashion but ACLs again will be changed.

```
# backup -f /dev/rmt0 -0 -u /:/user3
# restore -xf /dev/rmt0
```

We have also used SMIT backup and SMIT restore to backup files in a fileset and restore these files. The following two SMIT screens are provided to demonstrate our example. You can get to the SMIT screens by using the SMIT fast path commands of:

```
smit backup
```

and

```
smit restore
```

```
                        Backup Files in a Filesystem

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
    WARNING:  Execution of the backup command will
              result in the loss of all material
              previously stored on the selected
              output medium.

* DIRECTORY full pathname                            [/:/user3]
* Backup DEVICE or FILE                              [/dev/rmt0]
    (example: /dev/rfd0)
  REPORT each phase of the backup                     no                      +
  Backup local files only?                            no                      +
  Use data compression?                               yes                     +
  MAX. number of blocks to write on backup medium     []                       #



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Undo             F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

```
                        Restore Files in a File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Target DIRECTORY                                    [/:/user3]                /
  Restore DEVICE or FILE                              [/dev/rmt0]
  Number of BLOCKS to read in a single input          []                        #
    operation









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Undo             F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

5.5.5 Ensuring the File Systems are not Corrupted

The DCE DFS software has utilities that allow the file system to be checked for corruption and repaired if necessary. The following sections will discuss their operation.

5.5.5.1 Using the Salvager Program

Similar to the fsck program, which checks and repairs UNIX file systems, DFS offers the salvage program to maintain consistency on DCE LFS aggregates. The salvager program can only be used on DCE LFS aggregates. This program uses the log data (also called metadata) which are kept in the aggregate to record the structure and organization of a fileset. Each aggregate which holds LFS filesets keeps its own log data.

The salvage program does not make fsck obsolete. The fsck program is still applied to non-LFS filesets to check their consistency and eventually to repair them.

The salvage program has three main branches: the recover-only, the verify-only, and the salvage-only. These can be used to direct the salvager to:

1. Recover an aggregate: Recover an aggregate means to replay the log on an aggregate. This is the normal production use of the salvager program and this option is applied by default at system restart. Unless the log file or the physical structure of the aggregate is damaged this is an effective method of restoring the file system's integrity.
2. Verify an aggregate: Verifying the aggregate will check to see if the log mechanism is damaged or if the actual aggregate is damaged. This should be executed after the aggregate has been recovered.
3. Salvage an aggregate: Salvaging the aggregate will attempt to repair any inconsistencies. Salvaging the aggregate is what fixes the aggregate. If you have run the recovery and then the verification command and you were told that there are still inconsistencies, then you need to run the salvage command.

The salvage command invokes the DFS Salvager program on the aggregate specified with the -aggregate option. You must verify that the aggregate is detached (unexported) and also contains no locally mounted filesets before you start the salvager program on the aggregate. Use the following commands to perform this step.

```
# dfsexport -aggregate lfsusers3 -detach
# mount
```

Note that the -aggregate option for this command uses the physical device name as a parameter. In this example, lfsusers3 is the same as /dev/lfsusers3. This name is the local volume that was created with the mklv -y lfsusers3 -t lfs rootvg 1 command.

Let us first look at an example of the salvage program when we use the -recoveronly option to play back the log of metadata. This would be the standard use of the command:

```
# salvage -aggregate lfsusers3 -recoveronly -verbose
```

If you want to verify the consistency of the aggregate afterwards, then use the -verifyonly option.

```
# salvage -aggregate lfsusers3 -verifyonly -verbose
```

The -verifyonly option makes no changes to the aggregate and is useful if you want to assess the damage to an aggregate.

You may use both of the above in a single run, such as

```
# salvage -aggregate lfsusers3 -recoveronly -verifyonly -verbose
```

To attempt to repair any inconsistencies found in the structure of the aggregate use the -salvageonly option

```
# salvage -aggregate lfsusers3 -salvageonly -verbose
```

The salvager program will prompt the issuer only in rare situations, for example when it believes that there are non-LFS filesets in the aggregate or when the size of the aggregate exceeds the logical volume size. It also never asks the issuer on directions on how to repair a LFS fileset, and it can therefore run in the background and the process can be started multiple times simultaneously on different aggregates.

A successful execution of the salvager program with the verbose option on the aggregate lfspeter, which is not broken at the time looks similar to the following:

```
# salvage -aggregate lfspeter -recoveronly -verbose
Will run recovery on lfspeter
 SectorSize 512; TotalBlocks 511; BlockSize 8192;
 FragmentSize 1024; FirstBlock 1; NLogBlocks 13;
 NumBigChunks 65; minBlkSize 512; minBlkCount 8192;
 FileSysCreateTime 741913795;
 FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ''
/dev/rlfspeter: Episode file sys created Mon Jul  5 18:09:55 1993
Device /dev/rlfspeter, major 10, minor 13; total 511 8192-byte blocks.
Block size 8192, frag size 1024, firstBlock 1, nBlocks 511
Principal superblock at byte 65536, nLogBlocks 13.
```

Note

It is important to understand that file system integrity does not imply user data integrity. The salvage program can verify that each block in a file system is correctly attached, but the data in a block may be bad and the salvage program will not know about it. The owner of files have to verify the integrity of data themselves. This can be done by comparing the checksum (use the sum or the cksum command) of the file with that from a backed up copy of the file.

5.6 Cache Management

The DFS Cache Manager runs on all DFS client machines. The task of the DFS Cache Manager is to fetch and cache data from the DFS File Exporters and to make the data available for fast local access. The DFS Cache Manager must also evaluate if the data in the cache are still valid. This is done by keeping track of tokens which are sent from the File Exporter along with each file. The DFS Cache Manager is configured with the DFS client machine and initialized when the DFS client machine boots.

The cache is a fixed size piece of memory space or disk space which by default is located in the /var/dce/adm/dfs/cache directory of the local JFS file system.

This directory, called the cache directory is organized in slices (called chunks or V-files). Each chunk holds an entity of data (from 0 to 64 kilobytes), which is managed by the DFS Cache Manager.

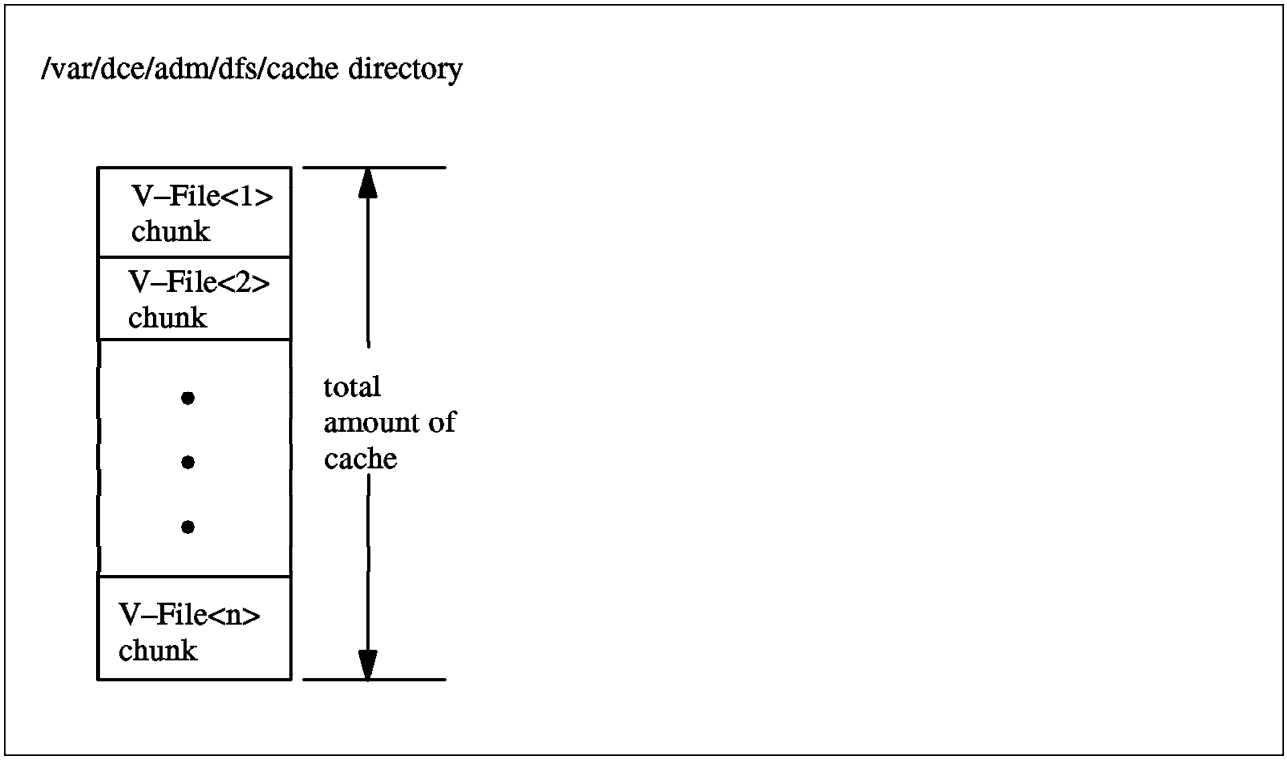


Figure 50. DCE DFS Cache

The size of the total cache and the size of the chunks are customizable and are important factors which can influence the performance of a machine heavily. The system administrator has no direct influence on how the DFS Cache Manager handles tokens but he can take action on the size, location and the chunk size of the cache.

The initial size of the cache is defined in a configuration file located at /opt/dcelocal/etc/CacheInfo. Changes made to this file will not be effective until DFS is rebooted. Make sure that the file system which houses the cache directory has enough space to maintain the cache. It is recommended that you create a new JFS filesystem for the cache and mount it at /var/dce/adm/dfs/cache. 10 Megabytes is the default for the cache size but this is easy to change to a different size during operation through a special command directed to the DFS Cache Manager or at DFS configuration time by editing the CacheInfo file. The CacheInfo configuration file created manually during DFS Client installation is composed of three fields:

1. The mount point for the root of the DFS filespace (usually /...)
2. The cache directory (default: /var/dce/adm/dfs/cache)
3. The size of the cache in 1k blocks (default: 10000)

Note that these fields are separated by a colon (:). Let us look at an example where the cache has 10 Megabytes and is located in the default directory /var/dce/adm/dfs/cache.

```
# pg /opt/dcelocal/etc/CacheInfo
/...:/var/dce/adm/dfs/cache:10000
```

The cache directory holds a large number of V-files and two control files. The V-files are the cache chunks fetched and stored back from the File Exporter. The control files are CacheItems and FilesetItems. Both of these are used to keep information for recovery in case of a restart. You should never try to change any files in the cache directory. If you accidentally do so, then you probably have to reboot the system to clean up.

Once the DFS Cache Manager is up and running it performs a number of tasks periodically:

- Every 30 seconds modified cache chunks are returned to the File Server
- Every 60 seconds unneeded tokens are returned
- Every 60 seconds CacheItems file is updated with status
- Every 60 seconds delayed writes for more than two minutes are checked
- Every 3 minutes servers thought to be down are checked to see if they are up
- Every 10 minutes servers thought to be up are checked to see if they are down
- Every 55 minutes the access times of files are stored back to the File Server
- Every 60 minutes all cached mount points are marked to be reevaluated
- Every 60 minutes keep-alive messages are sent to all File Servers

After the DFS Cache Manager is configured and set up initially, the system administrator may perform the following tasks:

- Monitor the usage of the cache
- Change the cache size temporarily
- Change the cache size and chunk size permanently
- Validate the functional operation of the DFS Cache Manager

The following section details how these tasks can be performed.

5.6.1 How to Monitor the Usage of the Cache

A helpful command to check if the DFS Cache Manager has contact to all File Servers of which it holds data, use the following command:

```
# cm statservers -all
```

When a File Server machine does not respond to this probe, the DFS Cache Manager marks the machine as nonfunctioning and it will display the name of the non-functioning file server.

The number of used blocks (1 kilobytes each) and the number of total available blocks in the cache directory can be displayed with the `cm getcachesize` command:

```
# cm getcachesize
DFS is using 400 of the cache's available 10000 1K byte (disk) blocks
```

The amount of total cache size may disagree with the number defined in the CacheInfo file. This happens, if the cache size was temporarily altered with the `cm setcachesize` or changed with the `dfsd -blocks` command.

If the number of used blocks you get from the cm getcachesize command is consistently very small compared to the total number of blocks in the cache, you may be able to improve the performance by increasing the number of chunks and using smaller chunk sizes.

Large chunks are useful if users work with large files, and small chunks are better for small files. A chunk can only hold data for one file at a time and if you have large chunks filled with small files there is a certain amount of disk space wasted.

5.6.2 How to Change the Cache Size Temporarily

The cache size can be changed temporarily without rebooting the machine with the cm setcachesize command.

```
# cm setcachesize -size 20000
```

This command changes the cache size to 20000 blocks (each with 1 kilobyte). If you specify a value of 0 for the number of blocks, the cache size will reset to the value from the CacheInfo file upon a subsequent reboot. A value of less than 5 for the number of blocks (less than 5 kilobytes) is not sufficient for convenient operation for disk caching. The default value is 10 Megabytes. If you have multiple users or if you rely heavily on cached information it is not uncommon to set this cache size higher.

Warning

Before you execute the setcachesize command to alter the cache size make sure that the JFS file system (normally /var) which houses the cache has enough space left, by issuing the df command.

IMPORTANT NOTE

It is highly recommended that a file system is defined solely for the use of the cache. This is so that a system administrator can determine the exact cache usage. This is also recommended so that the space allocated to the cache will not inadvertently get filled.

Also note that the logical volume that the cache resides in must be 15% larger that the size of the cache that you have defined in the /opt/dcelocal/etc/CacheInfo file. If this is not the case, you will experience abnormal system behavior when your cache gets full.

5.6.3 How to Change the Cache Size Permanently

The easiest way to change the size of the cache permanently is to edit the /opt/dcelocal/etc/CacheInfo file and set the number of blocks to the desired value.

```
/...:/opt/dcelocal/var/adm/dfs/cache:20000
```

This will set the cache to 20 Megabytes (20000 blocks, each with 1 kilobyte) at the next DFS restart.

Other methods of changing the cache size are available with options to the dfsd process. The dfsd daemon is activated from the /etc/rc.dfs shell script. It is

possible to provide options to this daemon by editing the /etc/rc.dfs file. The last lines of this file look like:

```
configured_daemons()
# Dummy function just to make sure these lines are not executed.
{

# The mkdce and mkdfs commands uncomment the appropriate daemons below
# (adding appropriate options) as the machine is configured.

daemonrunning $DCELOCAL/bin/rpcd
daemonrunning $DCELOCAL/bin/secd
daemonrunning $DCELOCAL/bin/sec_clientd
daemonrunning $DCELOCAL/bin/cdsadv
#daemonrunning $DCELOCAL/bin/cdsd
#daemonrunning $DCELOCAL/bin/gdad
daemonrunning $DCELOCAL/bin/dtsd
daemonrunning $DCELOCAL/bin/bosserver
daemonrunning $DCELOCAL/bin/dfsbind
daemonrunning $DCELOCAL/bin/fxd -mainprocs 7 -admingroup subsys/dce/dfs-admin
daemonrunning $DCELOCAL/bin/dfsd -blocks 20000

}
```

In this example, we have changed the highlighted line where the dfsd daemon is started. The - blocks 20000 option overwrites the CacheInfo configuration file and sets the total cache size to 20000 blocks (1K each). There are other options available for the dfsd daemon to change the behavior of the DFS Cache Manager. For example, you can set the chunk size or the number of chunks made available. Use the following option on the dfsd process to change the chunk size to 16 kilobytes (2**14) and change the number of chunks to 2000.

```
daemonrunning $DCELOCAL/bin/dfsd -files 2000 -chunksize 14
```

The dfsd process not only sets up cache configuration parameters. It is also used for prefetching of data and performing background I/O. On the dfsd process, the -mainprocs option can be used to specify the number of dfsd daemons running. Two daemons is the default. The number should be increased if you have more than five users working on the DFS client. This should result in an increase in performance. See the example below:

```
daemonrunning $DCELOCAL/bin/dfsd -mainprocs 6
```

5.6.4 Discarding Cache Data

If the DFS Cache Manager loses contact with a File Server System because of a network problem or a File Server being down, the DFS Cache Manager is unable to store the data from the cache to the File Server Machine. When this happens the user gets a message on the screen that warns him that the cache manager cannot write data back to the file server. The DFS Cache Manager tries to keep the data as long as possible but there are two cases when the DFS Cache Manager will discard this unstored data. They are:

- When the DFS Cache Manager needs the cache space for other data.
- When the user issues the cm resetstores command. This command tells the Cache Manager to discard all the data that it has been trying to store to the File Server which it cannot contact.

It is very unlikely that the DFS Cache Manager will discard the data by itself. However, it may happen from time to time that DFS File servers are down and

that the DFS Cache Manager continuously tries to reach these File Servers. To find out about these situations, the system administrator can issue the cm lsstores command:

```
# cm lsstores
```

This command will generate a list of the filesets that the DFS client cannot write data back to. At this time, the administrator needs to correct the network problem and/or fix the file server. Unstored data that is discarded from the cache is lost forever. The system administrator can discard all cache data that cannot be written to a file server by using the cm resetstores command.

```
# cm resetstores
```

5.6.5 Flushing Cached Data

When all chunks of the cache are in use and more space is required, the DFS Cache Manager needs to make decisions about which chunks are cleared for reuse. The strategy is to reuse chunks which have been stored in the cache longest but not used (LRU) and which hold reproducible data (such as from readonly filesets). You can force the DFS Cache Manager to flush the cache for single files or directories with the cm flush command or for full filesets with the cm flushfileset command. The appropriate commands are:

```
# cm flush -path /:/user3/peter/file1
# cm flushfileset -path /:/user3/peter/file1
```

Flushing a file, directory or fileset will not discard any changes made in the cache. All data are stored at the File Server first before reusing the cache.

5.6.6 Updating a Backup Version of a Fileset

The backup version of a fileset does not involve tokens when it is cached. Consequently the DFS Cache Manager would not recognize if a new version of the fileset has been created at the File Server. This problem is solved by a periodic check (once an hour) from the DFS Cache Manager for such situations. If you want the DFS Cache Manager to act immediately on such a condition, you must issue the cm checkfilesets command. Let us look at an example: Suppose, you remove a file from an LFS fileset and run an fts clone operation immediately after this, then you would expect that the file is also removed from your backup. As you can see from the following example this does not happen; the cached version still holds the file. You need to run the cm checkfilesets command to update the cache information. Note: The example assumes that the backup version is mounted at .Oldfiles.

```
# cd /:/user3/andrea
# ls
  .Oldfiles file1
# rm file1
# fts clone user3.andrea
# ls .Oldfiles
  .Oldfiles file1
# cm checkfilesets
# ls .Oldfiles
  .Oldfiles
```

5.6.7 Handling setuid Programs

A program with setuid permission allows other users to run this program with permissions that they might not normally have. These programs execute with the permissions of the owner of the program. The DFS Cache Manager by default does not allow setuid programs to execute with setuid permission. This would be a security risk and is therefore disabled. However the DFS system administrator can enable setuid permissions for selected filesets with the ″cm setsetuid″ command. Use the following commands to determine whether the DFS Cache Manager allows setuid programs:

```
# cm getsetuid -path /:/user3/peter
```

And use the following command to enable the setuid permission setting for selected filesets.

```
# cm setsetuid -path /:/user3/peter -state on
```

Be aware that setgid (group ownership) is not explicitly controlled by the DFS Cache Manager, but if the setuid is allowed on a fileset, the setgid is also allowed.

5.6.8 Enabling Access to Device Files in the DFS Filespace

By default, the DFS Cache Manager does not allow devices (or device files) to belong to the DFS filespace. For example, you could not use a tape drive whose special file /dev/rmt0 has been exported to the DFS filespace. If you want to allow the DFS Cache Manager to access device files that reside in a fileset in the DFS filespace, you must change the default setting to enable access to device files for a specified fileset. Device files are enabled on a fileset and on a cache-manager basis. This is commonly done at system startup in /etc/rc.

The following command queries the current status of the fileset where /:/user3/peter resides:

```
# cm getdevok -path /:/user3/peter
```

The following command enables access to device files that are located in the fileset that /:/user3/peter resides in for the cache manager on the local machine:

```
# cm setdevok -path /:/user3/peter -state on
```

Both of these commands work on complete filesets only.

Chapter 6. Security

This chapter explains the DCE Access Control Lists (ACLs) that are defined for DFS. The ACLs are used to control access to files and directories within DCE LFS filesets. They are used in conjunction with AIX mode bits. They have no relationship with AIX ACLs. The DCE DFS ACLs have a different format and are stored and controlled by the DCE DFS.

The chapter contains several examples that will better explain what ACLs are and how they are intended to be used.

6.1 What ACLs Are Used For

The DFS ACLs are used to specify which users and groups can have access to DFS files and directories. Traditionally, all UNIX systems have mode bits to implement protection of files and directories in a file system. The UNIX mode bits define permissions that are used to allow access of these files and directories. The type of access is typically read, write and execute.

Note that DCE DFS has two types of filesets. There are non-LFS (JFS) filesets and there are DCE LFS filesets.

The JFS filesets do not have any DFS ACLs associated with them. The files and directories in JFS filesets have the standard AIX mode bits and AIX ACLs to implement the protection of files and directories. Discussion in this chapter will pertain to files and directories in DCE LFS filesets unless otherwise noted.

The DCE LFS filesets have DFS ACLs in addition to the AIX mode bits. These DFS ACLs are useful in that they give DCE principals and groups some specific permissions. This type of granularity is what is really needed when machines are networked in a distributed environment. ACLs are used to extend the mode bits and make possible a more flexible and specific control of these objects.

By using ACLs, DCE DFS defines four permission types for files and six permission types for directories. They work as a supplement to the AIX mode bits. They do not replace them. The mode bits and the ACLs are always synchronized. This means that if you change the AIX mode bits, the change will also be reflected in the DCE DFS ACLs (and vice-versa).

6.2 How ACLS Work

An ACL for a DFS file may look like this:

```
# SEC_ACL for myfile:
# Default cell = /.../dino
mask_obj:rwx---
user_obj:rwxcid
user:cell_admin:r-x---
foreign_user:/.../saurus/ron:r-----
group_obj:rw----
other_obj:rwx---
```

This ACL shows several things. First it shows the filename and the default cell of the file called myfile. This implies that this is the default cell for user_obj,

group_obj, other_obj and any user and group ACL entries. All entries for foreign users (users from another cell) must explicitly use the name of their cell. This is shown for the foreign user named ron from the /.../saurus cell. The second thing that an ACL shows is an entry of the form:

type[:*key*]:*permissions*

The fields in this entry can be defined as follows:

The type is the particular kind of principal to which the permissions in this entry will apply. Various ACL entry types for users and groups are shown in Table 10 on page 163.

The key is a parameter that some types require. The key that names the particular principal or group to which the permissions in this entry will apply. An example of this is the entry for the user called cell_admin. An example of not using this parameter is for the user_obj. This specifies the owner of the file object.

The permission field defines the privileges that the principal/group that was named in the type and key fields are allowed to have. Read/Write/Execute (rwx) privileges are common privileges that UNIX users are familiar with. In addition, control, insert and delete (cid) are also defined. Please see Table 11 on page 166 for more information on how various operations on a file or directory are controlled by the permission bits.

The ACL operation is conceptually easy to understand. A user does a `dce_login` to authenticate his identity. The user then attempts to access a file or directory. The ACL for that file or directory is then checked in a particular order to see whether the user can do what he requested to that file or directory.

Here are some of the issues involved that tend to complicate the use of ACLs:

- How unauthenticated users get their resulting permissions
- Required permissions for requested operations on files and directories
- The order in which ACL entries are evaluated
- The effects of ACLs on the AIX mode bits
- ACLs are optional on files and directories in DCE LFS filesets
- How permissions are determined for locally mounted DCE LFS filesets
- How default ACLs are created for files and directories
- How permissions are determined for JFS filesets (no ACLs)
- How ACLs work for users in foreign cells

After establishing some basic concepts, we will investigate these issues in this chapter.

6.2.1 ACL Entry Types for Users and Groups

The following table lists the different types of ACL entries, their use of keys, and the users and groups to which they apply.

Table 10. ACL Entry Types for Users and Groups		
Type	**Key**	**Applies to**
user_obj	*none*	The user who owns the object
user	*user_name*	A specific user from the local cell
foreign_user	*cell_name/user_name*	A specific user from a specific foreign cell
group_obj	*none*	Members of the group that owns the object
group	*group_name*	Members of a specific group from the local cell
foreign_group	*cell_name/group_name*	Members of a specific group from a specific foreign cell
other_obj	*none*	Users from the local cell who do not match any of the above entries
foreign_other	*cell_name*	Users from a specific foreign cell who do not match any of the above entries
any_other	*none*	Users who do not match any of the above entries

ACLs for files and directories always contain a user_obj entry. This will have the same permissions as the AIX mode bits for the owner of a file (or directory). The ACL will also include a group_obj entry. This will have the same permissions as the AIX mode bits for group - if the mask_obj entry does not exist. If the mask_obj does exist, the AIX mode bits for group will be determined by the value defined by mask-obj. The last required entry in an ACL is the other_obj entry. This entry will have the same permissions as the AIX mode bits for other. All other entry types are optional.

6.2.1.1 The mask_obj Entry Type

The mask_obj ACL entry type is used by DCE DFS to maintain adherence to the POSIX standard for file and directory objects. It is used to filter (mask) the permissions granted by group_obj or specific group and user entry types. The mask_obj format is:

mask_obj:*permissions*

The mask_obj is logically ANDed with all entries in the ACL except user_obj and other_obj. This sets up a effective set of permissions that users may have. One purpose of mask_obj is to give an administrator some protection against assigning a user (or group) more permissions than originally intended. Therefore the user can be restricted to a maximum amount of permissions that is found in mask_obj.

The mask_obj is automatically created by DFS when there is an entry type in an ACL other than the three required types of user_obj, group_obj and other_obj. The mask_obj as well as any of the other ACL entry types can be changed by using the `acl_edit` command. Of course, you must be an authenticated user with proper permissions. The following examples will show how `acl_edit` is used and the effects it has upon mask_obj.

Notice that you can not delete mask_obj entry if there is a user (or group) type entry. In this case, when the commit (co) command is entered an error is returned and the ACL is not saved.

```
# acl_edit file1
sec_acl_edit> l

# SEC_ACL for file1:
# Default cell = /.../dino
mask_obj:rw----
user_obj:rwxc--
user:cell_admin:rw----
group_obj:rw----
other_obj:rwx---
sec_acl_edit>
sec_acl_edit> d mask_obj
sec_acl_edit> co
ERROR: not a valid DFS acl (dce / sec)
```

If you delete only the user entry, the ACL can be saved without the mask_obj entry.

```
# acl_edit file1
sec_acl_edit> l
# SEC_ACL for file1:
# Default cell = /.../dino
user_obj:rwxc--
user:cell_admin:rw----
group_obj:rw----
other_obj:rwx---
sec_acl_edit> d user:cell_admin
sec_acl_edit> co
sec_acl_edit> exit
```

Notice how the automatic mask_obj creation works. When you add the first user or group entry to an ACL, a mask_obj is automatically generated for that ACL.

```
# acl_edit file1
sec_acl_edit> l

# SEC_ACL for file1:
# Default cell = /.../dino
user_obj:rwxc--
group_obj:rw----
other_obj:rwx---
sec_acl_edit> m user:cell_admin:rx
sec_acl_edit> l

# SEC_ACL for file1:
# Default cell = /.../dino
mask_obj:rwx---
user_obj:rwxc--
user:cell_admin:r-x---
group_obj:rw----
other_obj:rwx---
sec_acl_edit> co
sec_acl_edit> exit
```

Note: The rule requiring user_obj, group_obj, other_obj entries, and the presence of mask_obj entry with an entry other than an _obj entry exists only for ACLs on DCE LFS objects, not for ACLs on objects associated with other DCE components. DCE LFS enforces these restrictions to maintain adherence to the POSIX** standard for ACLs on file and directory objects.

6.2.2 ACL Permissions

We have mentioned that the purpose of an ACL is to allow users and groups to have a certain set of permissions for a particular file or directory. We will now define what these permissions allow the user or group to do. Note that these six permissions are different from the permissions that are associated with objects in the DCE namespace. These permissions are for file and directory objects in the DFS namespace.

For a file, valid permissions can be:

- read (**r**)
- write (**w**)
- execute (**x**)
- control (**c**) (used to allow the user (or group) to change the ACLs on this file)

For a directory, valid permissions can be:

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

The insert and delete permissions basically allow a user (or group) to insert objects or delete objects (files or directories) from the specified directory.

Most of the time you need multiple permissions to work with files and directories. For example, operations performed on files and directories need the execute (X) permission on all directories that lead to the directory or file that you want to work with. Please refer to Table 11 on page 166 to see the permissions that are required in order to do operations on files and directories.

Limiting Access to Objects

It is possible to access an object even if you do not have execute (x) permission on every directory leading to that object. This is because the creation of mount points and hard links may circumvent the usual pathname transversal. The x permission of a parent directory should not be used as the only method of securing objects in a directory. (Use appropriate ACL settings on the object itself.)

Table 11. 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 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
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 • **x** on all directories that lead to the object

6.2.3 Order of ACL Evaluation

We have seen that an ACL can contain many entries. Whenever a user tries to access an object that is protected by an ACL, the ACL entries are searched from most specific to least specific entry types. The algorithm that is used by the DFS to allow a user access to an object is shown in Figure 51 on page 167. Please refer to that figure for the following discussion. Note that as soon as the user (or group) finds a match in the ACL, the evaluation stops right there and the user is assigned the corresponding permissions.

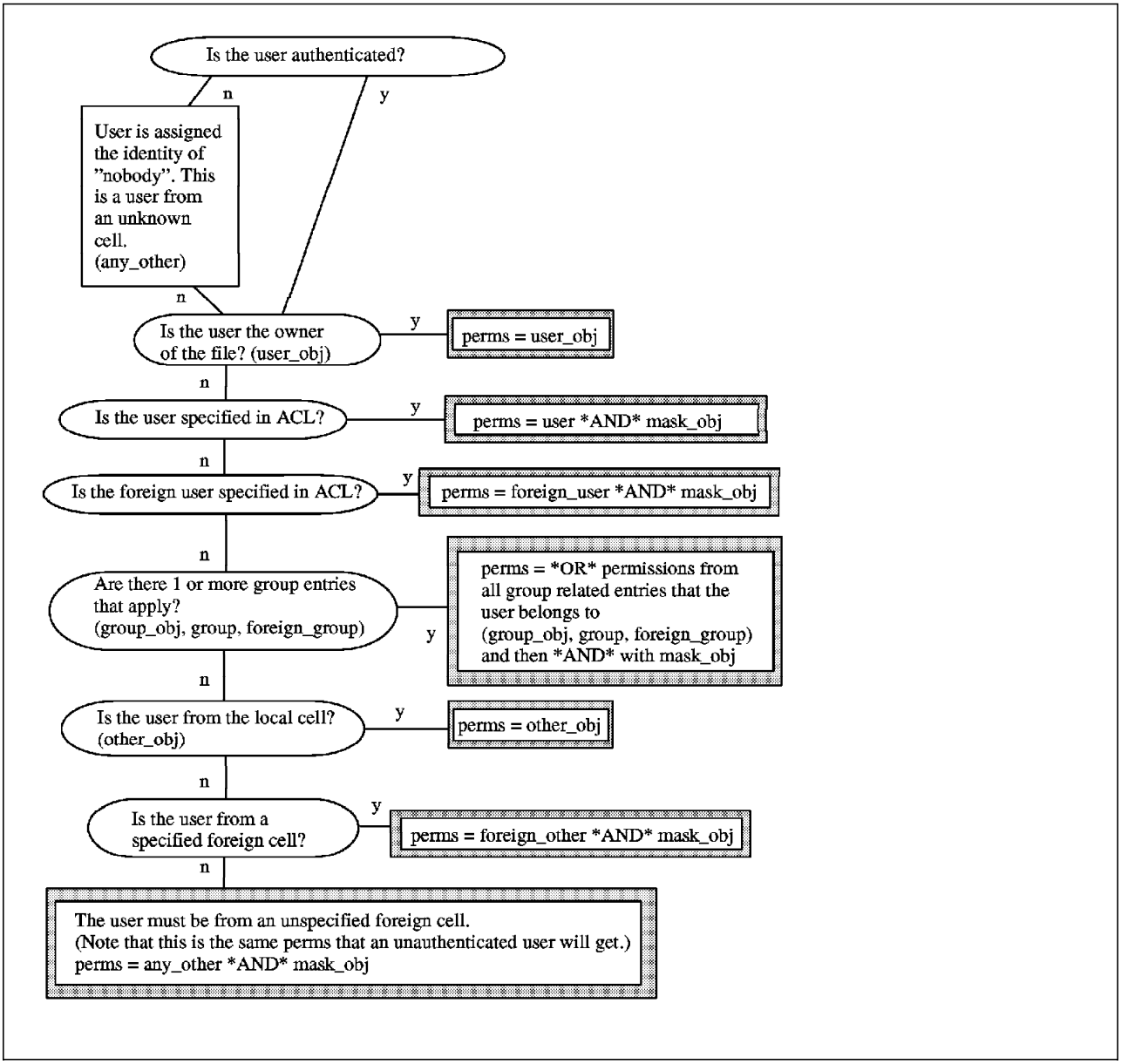


Figure 51. Order of ACL Evaluation

The order of ACL evaluation proceed as follows. The authenticated user is checked to see whether he is authenticated (logged in to DCE via `dce_login`). We will discuss the unauthenticated case in section 6.2.3.1, "ACLs and Unauthenticated Users" on page 168. The user is checked to see whether he is the owner of an object. If this is the case, he receives the permission from the user_obj entry and the ACL evaluation would stop at this point. If not, the user is checked for a type entry of user. If a user:principalname entry is found where principalname matches the user's name, the ACL evaluation is stopped and he receives the result of the AND of mask_obj and the permissions of the user:principalname entry.

The foreign_user types are checked in the same manner.

If the permissions for the user are still not known at this point, all the permissions from the group_obj, group and foreign_groups that the principal is a

member of are logically ORed together. This resulting permission is then ANDed with mask_obj and the ACL evaluation procedure is over.

If the user was not a member of any groups that were listed in the ACL, the evaluation proceeds. If the user is from the local cell, he receives the permissions associated with the other_obj entry. If the user is from one of the foreign cells that are listed in a foreign_other type entry, then he receives the AND of the mask_obj with the permissions for the foreign_other type entry.

If the user is from a foreign cell that is not listed in a foreign_other type entry he receives the permissions associated with the any_other type entry ANDed with the mask_obj. This is also where all unauthenticated users receive their permissions from.

As you can see, the order in which the types are checked determines the permissions for the object. This order is checked from most specific to least specific.

6.2.3.1 ACLs and Unauthenticated Users

A user that does a dce_login is called an authenticated user. DFS can use the ACLs to determine what permissions this user should have. If the user has not logged into DCE, he is an unauthenticated user. This could be an AIX user that logged into AIX but not into DCE. Therefore, DCE can not verify his true identity. The DFS evaluates an unauthenticated user's identity as follows: The DFS assigns the unauthenticated user the id of *nobody*. This is then treated as a user from an unknown cell that does not match any other entries. The nobody user will receive the permissions associated with the any_other entry.

Unauthenticated users must have any_other entry

To allow an unauthenticated user to gain access to DCE LFS files or directories the object's ACLs *must have the any_other entry defined. All parent directories that lead to the object must also have this entry defined with the r,x permissions.*

Unknown Cell Message Displayed Using acl_edit

When an unauthenticated user creates a DCE DFS object, its ACLs will have an unknown cell as a default cell. At the time you view or edit this ACLs, an error message will be displayed. It is NORMAL and the message can be ignored.

For non-LFS Filesets Only

Unauthenticated users and authenticated users from foreign cells are treated as the user *nobody* when they access an object in a non-LFS fileset. As a result, they receive the permissions associated with the *other* AIX mode bits.

6.2.3.2 ACL Evaluation for Locally Mounted Filesets

It is possible to locally mount a DCE LFS fileset as a file system on the File Server machine. A local mount is simply the result of using the AIX `mount` command to mount the fileset. You can access an object in the DCE LFS fileset that is mounted locally using two different paths: one is the local path, and the other is a DCE path.

Recommendation

It is recommended that you only use local mounting to perform emergency system administration.

The `mount` command looks as follows:

```
# mount -v lfs -o aggregate=<aggregatename> -n
<hostname> <filesetname> <directory on which to mount>
```

It is necessary to indicate the local hostname with the `-n` node parameter. The `mount` command will return without an error if this parameter is not used - but the mount will not really happen.

The same ACL evaluation algorithm is used to determine your permissions in either case. The permissions you receive when you access an object by the way of its local path are those associated with your local identity - in other words, the UID (AIX user ID) that you used when you did a login to AIX. When you use the DCE path the permissions you receive are those associated with your dce_login identity. This is also true for objects in non-LFS filesets.

Figure 52 on page 170 shows the algorithm that is used to determine which principal identity (dce login identity or AIX login identity) is used for checking ACL permissions.

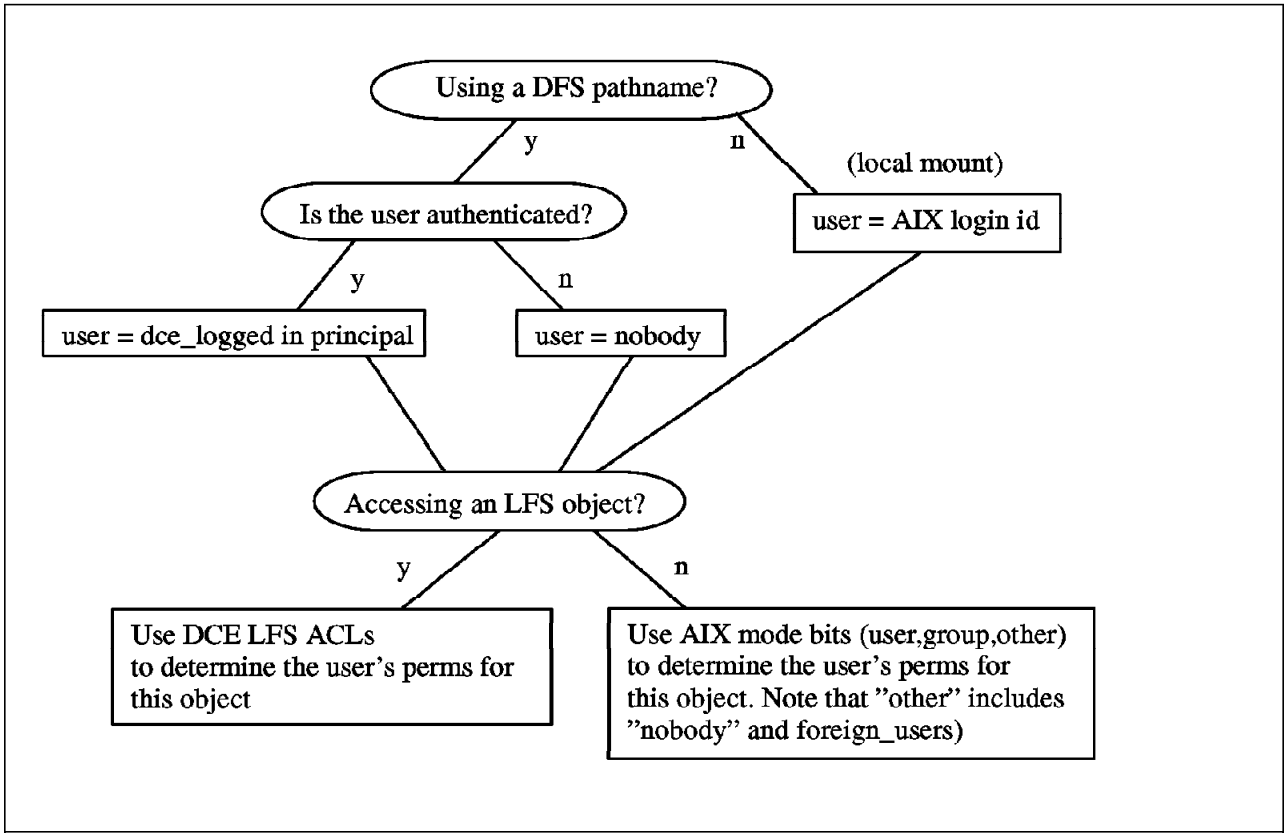


Figure 52. Accessing Files in Local and DCE Mounted File Systems

The following is an example of showing how files in the DFS can be accessed in multiple ways if they are both DCE mounted and also locally mounted.

The LFS fileset user1.fs is local mounted in the directory /test_loc.

The DCE DFS path for the fileset's directory is /:/user1.

```
# dce_login user1 -dce-
Password must be changed!
# cd /:/user1
# mkdir test1
# acl_edit test1
sec_acl_edit> l

# SEC_ACL for test1:
# Default cell = /.../dino
user_obj:rwxcid
group_obj:r-x---
other_obj:r-x---
sec_acl_edit> m user:user1:rwxid
sec_acl_edit> l

# SEC_ACL for test1:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
user:user1:rwx-id
group_obj:r-x---
other_obj:r-x---
sec_acl_edit> commit
```

```
sec_acl_edit> e
# dce_login user1 -dce-
# id
uid=201(takeda) gid=1(staff) groups=12(dce)
# cd /test_loc/test1
# ls -al
total 4
drwxrwxr-x   2 guest    dce          256 Jul 26 14:35 .
drwxrwxr-x   4 root     system       384 Jul 26 14:35 ..
# touch newfile
touch: 0652-046 Cannot create newfile.
```

Notice that a file (newfile) could not be created. This is because we are accessing this file with an AIX pathname. The result is that the AIX login id is used. In our example, this is takeda and the user takeda does not have permission to write into the test1 directory.

```
# pwd
/test_loc/test1
# ls -al
total 4
drwxrwxr-x   2 guest    dce          288 Jul 26 14:38 .
drwxrwxr-x   4 root     system       384 Jul 26 14:35 ..
```

If the user uses the DCE DFS pathname the system associates his identity to the DCE principal identity (user1) and uses DCE DFS ACLs to verify the permissions. The ACLs grant the DCE user1 principal permissions to read, write, insert and delete objects in the directory. Thus, the creation of the file is possible.

```
# cd /:/user1/test1

# ls -al   total 4
drwxrwxr-x   2 guest    dce          320 Jul 26 14:57 .
drwxrwxr-x   4 root     system       384 Jul 26 14:35 ..
# touch newfile
# ls -al
drwxrwxr-x   2 guest    dce          288 Jul 26 14:38 .
drwxrwxr-x   4 root     system       384 Jul 26 14:35 ..
-rw-r--r--   1 124      dce            0 Aug  5 14:36 newfile
# acl_edit /:/user1/test1
sec_acl_edit> l

# SEC_ACL for /:/user1/test1:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
user:user1:rwx-id
group_obj:r-x---
other_obj:r-x---
sec_acl_edit> ab
# id
uid=201(user1) gid=1(staff) groups=12(dce)
# klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../dino/user1
        Cell:      007979f8-7d23-1c38-9eb1-10005aa83bfa /.../dino
        Principal: 0000007c-2856-2c54-9600-10005aa83bfa user
        Group:     0000000c-7d23-2c38-9e01-10005aa83bfa none
        Local Groups:
```

```
                    0000000c-7d23-2c38-9e01-10005aa83bfa none

Identity Info Expires: 93/07/27:00:38:22
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_414f2683
Default principal: user@dino
Server: krbtgt/dino@dino
        valid 93/07/26:14:38:22 to 93/07/27:00:38:22
Server: dce-rgy@dino
        valid 93/07/26:14:38:22 to 93/07/27:00:38:22
Server: dce-ptgt@dino
        valid 93/07/26:14:57:59 to 93/07/26:16:57:59
Client: dce-ptgt@dino   Server: krbtgt/dino@dino
        valid 93/07/26:14:58:00 to 93/07/26:16:57:59
Client: dce-ptgt@dino   Server: hosts/sys1/dfs-server@dino
        valid 93/07/26:14:58:00 to 93/07/26:16:57:59
Client: dce-ptgt@dino   Server: dce-rgy@dino
        valid 93/07/26:14:59:42 to 93/07/26:16:57:59
```

6.2.4 ACL Inheritance

ACL inheritance is the method by which an object is given an ACL when it is created. As shown in Figure 53 on page 174, there are two kinds of objects that can be created. There is a file object and a directory object. This figure shows that certain default values are passed to files and directories from their parent directory. For files, the values that are passed are the Initial Object Creation (IOC) ACL. For directories, the values that are passed are the Initial Container Creation (ICC) ACL and the IOC.

The DCE documentation sometimes refers to directories as container objects. This is because these directories contain files (also called file objects). The IOC and the ICC values are just default ACLs that are used when creating new files and directories.

There are three things to consider when the ACL is created for the new object:

1. AIX mode bits specified with the creat, open or mkdir system call (The access granted will never be more permissive than indicated by the mode parameter specified in the creating interface.)
2. Initial Creation ACL from the object's parent directory
3. umask of the process that creates the object

If the parent directory has an Initial Creation ACL defined, only items 1 and 2 are considered. The umask of the process that creates the object is not used.

If the parent directory does not have an Initial Creation ACL defined, only steps one and three are considered. This is the same procedure as a regular file getting created under AIX (and not DFS). In this case, an ACL for the newly created file or directory will *not* be created and the protection will be only by AIX mode bits.

The following list summarizes the ACL inheritance for files when an Initial Creation Ovject ACL exists:

1. file's **user_obj**

- **mode** (user rwx bits) & IOC **user_obj** (rwx bits)
- copy IOC **user_obj** c bit to file's **user_obj** c bit

2. file's **group_obj**
   - if IOC **mask_obj** DOES NOT exist
     - **mode** (group rwx bits) & IOC **group_obj** (rwx bits)
     - copy IOC **group_obj** c bit to file's **group_obj** c bit
   - else
     - mode (group rwx bits) & IOC mask_obj (rwx bits)
     - copy IOC **group_obj** c bit to file's **group_obj** entry
3. file's **other_obj**
   - **mode** (other rwx bits) & IOC **other_obj** (rwx bits)
   - copy IOC **other_obj** c bit to file's **other_obj** c bit
4. file's **mask_obj**
   - if IOC **mask_obj** DOES NOT exist
     - file's **mask_obj** is not defined
   - else
     - **mode** (group rwx bits) & IOC **mask_obj** (rwx bits)
     - copy IOC **mask_obj** c bit to file's **mask_obj** c bit
5. All other entries that exist in the parent directory's IOC ACL are copied to the file's Object ACL.

DCE LFS uses the same algorithm to determine the initial entries and permissions for a subdirectory's Object ACL, using the parent directory's ICC ACL instead of its IOC ACL. The subdirectory also inherits its parent's ICC ACL as its ICC ACL, and it inherits its parent's IOC ACL as its IOC ACL.

To see the ACL inheritance for subdirectories (as opposed to files), substitute IOC by ICC in the previous list.

The following figure illustrates ACL inheritance for files and directories.

Note

DCE ACL inheritance will not cross a fileset mount point. This means that if a fileset is mounted at /:/dir1, then a file that is created in this /:/dir1 directory will have permissions influenced in part by the ACL that you would see when you use the `acl_edit /:/dir1 -io` command. If another fileset were to be mounted below it at /:/dir1/dir2 and you created a file, then the ACL that this file would receive would be influenced in part by the ACL that you would see with the `acl_edit /:/dir1/dir2 -io` command.

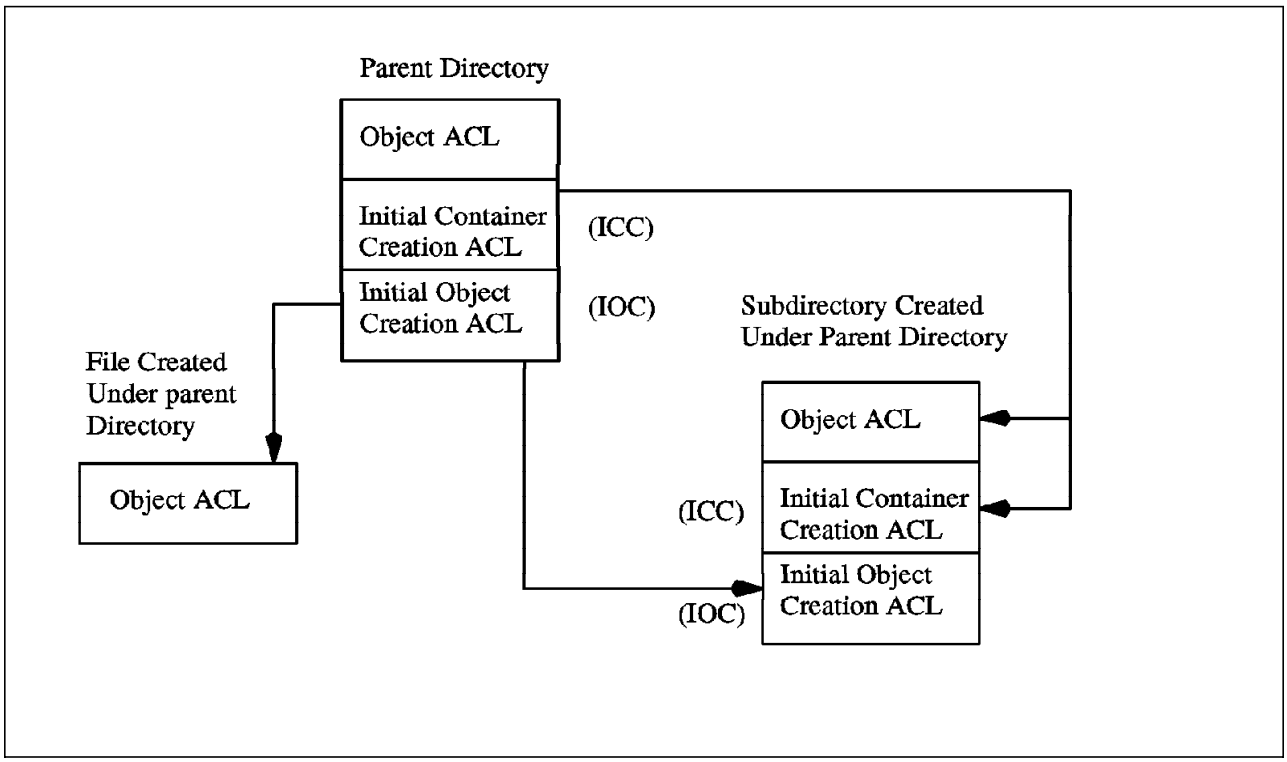


Figure 53. ACL Inheritance

6.2.4.1 Using acl_edit to Set and Examine DCE LFS ACLs

The `acl_edit`. command from the DCE Security Service is used to list and modify the ACLs of a DCE LFS object. The command can be used in interactive or command-line mode. This section documents some DFS-specific information about the command and provides a brief example of its use. Refer to the online documentation provided by InfoExplorer for complete details about the command syntax and usage.

In most respects, the operation of the `acl_edit` command with DCE LFS objects is the same for other types of DCE objects. To examine an ACL for a file or directory you must have the execute permission on the directory in which the object resides, as well as on all directories that lead to that directory. To modify an entry in an ACL for a file or directory you must have the control permission for the object, as well as the execute permission on all directories that lead to the object.

See section 6.2.2, “ACL Permissions” on page 165 for more information about permissions.

WARNING: acl_edit save function

Be especially careful when using the acl_edit command in interactive mode. The command allows you to violate restrictions as long as you remain in interactive mode. However, it does not let you save the ACL with the commit or exit subcommand. Moreover, if you violate a restriction in the process of making other, valid changes, the command discards *all* changes when you use the exit subcommand in an attempt to save your changes and leave acl_edit. It is always better to use the commit subcommand to save changes to an ACL. If you violated a restriction, the command reports that it cannot save the ACL but it does not discard your changes.

You can use acl_edit to view/modify/insert/test ACL entries of a DCE LFS object. To invoke the acl_edit command in interactive mode to work in the object's Object ACL, you can use the following example:

```
# acl_edit filename
```

At the prompt of acl_edit command you can see all the subcommands it accepts and their shortcuts, using **h**elp subcommand:

```
sec_acl_edit> h
Known commands are:
        ab[ort]          as[sign_file]   d[elete]        ce[ll]
        co[mmit]         g[et_access]    e[xit]          h[elp]
        k[ill_entries]   l[ist]          m[odify]        pe[rmissions]
        pu[rge]          sec_acl_entry   su[bstitute]    t[est_access]
        ?
sec_acl_edit>
```

To list the Object ACL of the DCE LFS object, use the list command:

```
sec_acl_edit> l

# SEC_ACL for filename:
# Default cell = /.../dino
user_obj:rw-c--
group_obj:r-----
other_obj:r-----
sec_acl_edit>
```

You can see in this example that the DCE LFS object has only the mandatory ACL entries (user_obj, group_obj, and other_obj), also, you can see the format of the permissions list. They appear in the sequence **r**ead, **w**rite, e**x**ecute, **c**ontrol, **i**nsert, and **d**elete. If the permission is not present, its position is marked with a dash (-).

To modify an entry, use the subcommand **m**odify:

```
sec_acl_edit> m user_obj:rwxcid
sec_acl_edit> l

# SEC_ACL for filename:
# Default cell = /.../dino
user_obj:rwxcid
group_obj:r-----
other_obj:r-----
sec_acl_edit> m group_obj:rwx
sec_acl_edit> l
```

```
# SEC_ACL for filename:
# Default cell = /.../dino
user_obj:rwxcid
group_obj:rwx---
other_obj:r-----
```

To create a new entry you also use the **m**odify subcommand:

In this example we are creating a mask_obj entry.

```
sec_acl_edit> m mask_obj:r
sec_acl_edit> l

# SEC_ACL for filename:
# Default cell = /.../dino
mask_obj:r-----
user_obj:rwxcid
group_obj:rwx---        #effective:r-----
other_obj:r-----
sec_acl_edit>
```

Notice that after the introduction of the mask_obj entry, the group_obj entry appears different. It reflects the masking operation, showing the effective protection that is granted the users that match this entry.

The entries until now have not used the key tag. To introduce an entry for a specific user, do as follows:

```
sec_acl_edit> m user:cell_admin:r
sec_acl_edit> l

# SEC_ACL for filename:
# Default cell = /.../dino
mask_obj:r-----
user_obj:rwxcid
user:cell_admin:r-----
group_obj:rwx---        #effective:r-----
other_obj:r-----
sec_acl_edit>
```

For a user from a different cell:

```
sec_acl_edit> m foreign_user:/.../sauro/cell_admin:r
sec_acl_edit> l

# SEC_ACL for filename:
# Default cell = /.../dino
mask_obj:r-----
user_obj:rw-c--
user:cell_admin:r-----
foreign_user:/.../sauro/cell_admin:r-----
group_obj:rwx---        #effective:r-----
other_obj:r-----
sec_acl_edit>
```

The following is a summary of the other subcommands you may use with the acl_edit command:

- The **ab**ort subcommand is used to exit acl_edit without saving the changes.

- The **as**sign subcommand enables to copy ACL entries of another file to the specified object. It replaces existing entries with the copied entries.
- The **ce**ll subcommand sets the cell name to be assigned to the principal or group in place of the local cell name. This subcommand is used to facilitate copying ACLs from different cells.
- The **co**mmit subcommand is used to make the changes to the ACLs. Use this subcommand to test the changes you made so far and also to prevent the lost of changes that may be discarded when you exist with illegal modifications to the entries.
- The **d**elete subcommand deletes a specified ACL entry.
- The **g**et_access subcommand displays the permissions granted in the specified object's ACL to the principal that invoked **acl_edit**.
- The **k**ill_entries removes all ACL entries except the **user_obj** entry if it exists.
- Use **pu**rge subcommand to purge all masked permissions.
- The **pe**rmissions subcommand gives a list of permissions and a brief summary of them.
- The **su**bstitute subcommand replaces all ACL entries with the one or ones specified. This subcommand removes all existing entries and adds the ones specified as arguments.
- The **t**est_access subcommand tests whether or not the permissions specified in the subcommand are granted to the principal under whose DCE identity the `acl_edit` command was invoked.

To view or modify a directory's Initial Object Creation (IOC) ACL you must use `-io` option when invoking `acl_edit` command. In the same way, to view or modify a directory's Initial Container Creation (ICC) ACL you must use `-ic` option when invoking `acl_edit` command.

Refer to section 6.2.4, "ACL Inheritance" on page 172 for complete description of IOC and ICC ACLs.

6.2.5 Initial ACLs of a New Fileset

When a DCE LFS fileset is first created (by using the `mkdfslfs` command), it is mounted into the DFS filespace. The permissions for the directory of this fileset are the AIX mode permissions. An ACL for this directory does not really exist at this time. If you use `acl_edit` on this directory, an ACL is created that is synchronized with the AIX mode permissions. In Figure 54 on page 178, it is shown what the permissions are like for a newly created DCE LFS fileset. We have created this fileset and mounted it as the /:/newlfs directory. The Initial Container Creation ACL and the Initial Object Creation ACL are also shown. (`acl_edit /:/newlfs -ic` and `acl_edit /:/newlfs -io` commands)

Advantage of using DCE ACLs

A DCE LFS fileset can include many files and directories that do not have ACLs. However this approach fails to take advantage of the enhanced security available with DCE ACLs. Therefore, it is important to use the `acl_edit` command to create the Object ACL, IOC ACL, and ICC ACL for the root directory of a fileset *before* other objects are created in the directory. In this manner subdirectories and files created under this DCE LFS directory will automatically inherit the ACLs (IOC ACLs for files and ICC ACLs for directories). You should also make sure that the owner and group is set appropriately for the root directory of a fileset.

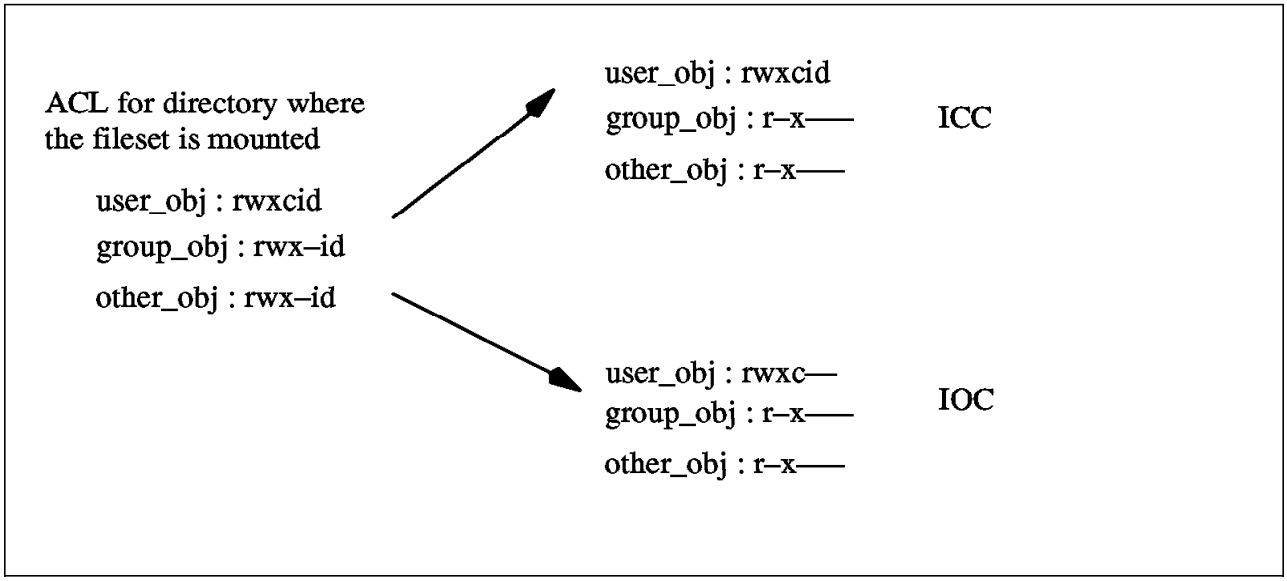


Figure 54. ACL Permissions for a New Fileset

You may wonder why the values of the ICC and IOC are as they are shown. This is because the DCE LFS uses the umask of the process that involved the creation of the fileset as well as the mode bits on the creat or open system call that created this directory. The open/creat system call is done *under the covers* when the fileset is created so it is not obvious to the user as to what the resultant permission bits will be for the IOC and ICC ACLs.

Future Change for Simplification Purposes

It is expected that in future releases that the default ICC will be set to 755 (rwxr-xr-x). The default IOC will be set to 755 (rwxr-xr-x).

6.2.5.1 Setting Up an DCE LFS Fileset For a User

One of the tasks that a cell administrator will do from time to time is to set up an DCE LFS fileset for a user. In most cases, the use of ACLs is highly recommended.

Here is an example of how this is done. First add a new logical volume:

```
# mklv -y testlfslv -t lfs rootvg 1
testlfslv
```

Next, create a new aggregate on that logical volume.

```
# newaggr -aggregate /dev/testlfslv \
>  -blocksize 8192 -fragsize 1024 -overwrite
```

Note

The -t option in the mklv command is used to add a label for the logical volume that is created. If we do not put a label on and we leave it blank, the system will assume a JFS file system as the default. You can use the following command to verify that an DCE LFS file system has really been created (and labeled properly).

```
# lsvg -o |lsvg -il
```

We will mount this into the DFS namespace at /:/test1.

```
# mkdfslfs -f test1 -m /:/testlfs -d /dev/testlfslv -n testaggr

# cd /:

# li -l
Drwxrwxrwx    2 root     system        256  Nov 15 18:11 testlfs
```

Note the default permissions that were created for this directory. If we look at the ACL that was implicitly created, we will see that the permissions will be rwx for user_obj, group_obj and other_obj.

```
# acl_edit /:/testlfs
sec_acl_edit> l

# SEC_ACL for /:/testlfs:
# Default cell = /.../saurus
user_obj:rwxcid
group_obj:rwx-id
other_obj:rwx-id
```

At this time, we will allow the ownership of this directory to be transferred to a user called ron. The user ron will now have the user_obj permissions that allow him to change the information in the ACL. (The user_obj has the **c** permission bit set.) The ownership of this ACL is changed with the AIX chown command. You could also change the group ownership with the AIX chgrp command.

```
# chown ron /:/testlfs
# li -l
Drwxrwxrwx    2 ron      system        256  Nov 15 18:11 testlfs
```

At this time, ron now has the rwxcid permissions that were associated with user_obj.

Now would be the appropriate time for ron to modify the IOC and ICC ACLs to accept specified users and groups if ron wanted to propagate that change to all files and directories that are created after this. At this point, we will just show what the IOC and ICC ACLs contain.

```
# acl_edit /:/testlfs -io
sec_acl_edit> l

# Initial SEC_ACL for objects created under: /:/testlfs:
# Default cell = /.../saurus
user_obj:rwxc--
group_obj:r-x---
other_obj:r-x---
sec_acl_edit>exit

# acl_edit /:/testlfs -ic
sec_acl_edit> l

# Initial SEC_ACL for directories created under: /:/testlfs:
# Default cell = /.../saurus
user_obj:rwxcid
group_obj:r-x---
other_obj:r-x---
sec_acl_edit>exit
```

6.2.6 Difference Between DCE LFS ACLs and AIX ACLs

Until now, we only described the DCE LFS ACLs. Do not forget that AIX has its own ACLs that can be used to control access to files and directories in the AIX file system.

The acl_edit command is used to edit DCE LFS ACLs. There are no DCE LFS ACLs defined for non-LFS filesets files and directories.

AIX ACLs are used to extend the AIX mode bits (base permission bits) to permit or deny access to specific users or groups, without changing the base permissions. You use acledit to enable the AIX ACL extended permissions and edit individual entries to permit, to deny, or to specify permissions. See the explanation of Access Control Lists in IBM InfoExplorer for more details.

Example of AIX ACLs:

```
  attributes: SUID
  base permissions:
        owner(root):  rw-
        group(system): r-x
        others: ---
  extended permissions:
        enabled
          permit  rw-  u:takeda
          deny    r--  u:ron, g:staff
          specify r--  u:peter, g:system, g:mail
          permit  rw-  g:account, g:finance
```

You should never use acledit (the AIX command to edit AIX ACLs) on DCE LFS objects, but it will not affect any entry that is already defined. It also will not allow you to make changes in the object's DCE LFS ACLs. You can try to extend the permissions using AIX ACLs extended entries but they will not be saved to the object's ACLs. The AIX acledit command will try to work on DCE LFS objects without warning that you are using the wrong command.

Even when you try to use acledit command on files or directories that are local mounted, the effect will be the same. The only way to edit DCE LFS object's ACLs is by using the acl_edit command.

WARNING: use the CORRECT command

Be careful when typing the command for ACL editing. They are very similar and is easy to use acledit instead of acl_edit. Remember acl_edit for DCE LFS ACLS and acledit for the AIX ACLs that have nothing to do with DCE or DFS.

6.3 Security Considerations When Using DCE ACLs

In this section we provide some issues that may affect security of your system.

6.3.1 Authorization Problems Caused by Using UNIX (AIX) backup Commands

It is very common to use traditional AIX commands like tar, cpio, dd to copy files and directories and also to make backups. These commands do not know about DCE LFS ACLs and therefore do not save the ACL information.

See section 5.5.4.1, "Use of the tar Command" on page 148 for more details.

6.3.2 Authorization Problems Caused by Local Pathnames or DCE Pathnames

When accessing DCE LFS objects through local pathnames, using locally mounted filesets, the authentication mechanism will use the AIX user id instead of DCE principal ID. The user will be considered an unauthenticated DCE UID. foreign. The appropriate ACL entries will be used and you should have supplied the appropriate permissions in this entries.

6.3.3 AIX Group Mode Bits and DFS ACL Interaction Caused by mask_obj

The ACL evaluation and the ACL interaction with AIX mode bits can be very confusing. The following example of how they work should clarify the matter. Let us consider a file in an DCE LFS fileset named yourfile. Its ACLs show the following values:

```
# ls -l yourfile
-rw-r--r--   1 takeda   dce         2118 Jul 21 10:00 yourfile
```

Since the AIX mode bits are synchronized with the DFS ACL, we can show the DFS ACL that was implicitly created for the file. Note that mask_obj does not exist because there are no user or group types defined.

```
# acl_edit yourfile
sec_acl_edit> l
# SEC_ACL for yourfile:
# Default cell = /.../dino
user_obj:rw-c--
group_obj:r-----
other_obj:r-----
```

Let us now define a user type. Note that the mask_obj is implicitly created with the same permissions (rwx) as the user that we defined.

```
sec_acl_edit> m user:peter:rwx
sec_acl_edit> l
# SEC_ACL for yourfile:
# Default cell = /.../dino
mask_obj:rwx---
user_obj:rw-c--
user:peter:rwx---
group_obj:r-----
other_obj:r-----
```

Let us examine the effect of changing the group_obj as well as the mask_obj itself. As you now see, the user and the group_obj are now restricted by the mask_obj. They are logically ANDed with the mask_obj and this results in the creation of an effective set of permissions for the user and the group_obj.

```
sec_acl_edit> m group_obj:rwx
sec_acl_edit> m mask_obj:rx
sec_acl_edit> l

# SEC_ACL for yourfile:
# Default cell = /.../dino
mask_obj:r-x---
user_obj:rw-c--
user:peter:rwx---       #effective:r-x---
group_obj:rwx---        #effective:r-x---
other_obj:r-----
sec_acl_edit> e
```

Once again, the AIX mode bits are synchronized as follows:

- AIX user mode bits = user_obj
- AIX other mode bits = other_obj
- AIX group mode bits = mask_obj

(If mask_obj does not exist, then the AIX group mode bits = group_obj.)

```
# ls -l yourfile
-rw-r-xr--   1 takeda   dce         2118 Jul 21 10:00 yourfile
```

We again show that by modifying the mask_obj, it results in changing the AIX group mode bits.

```
# acl_edit yourfile
sec_acl_edit> m mask_obj:rwx
sec_acl_edit> l

# SEC_ACL for yourfile:
# Default cell = /.../dino
mask_obj:rwx---
user_obj:rw-c--
user:peter:rwx---
group_obj:rwx---
other_obj:r-----
sec_acl_edit> e
# ls -l yourfile
-rw-rwxr--   1 takeda   dce         2118 Jul 21 10:00 yourfile
#
```

Note that changing the AIX mode bits with the chmod command will cause the DFS to change the corresponding user_obj, other_obj and either mask_obj or group_obj entries in the ACL.

Be Careful When Modifying mask_obj

You must be careful when changing the DCE LFS ACL mask_obj entry because of its interaction with AIX group mode bits. These AIX group mode bits are formed by the DCE LFS ACL mask_obj entry if it exists or by the group_obj entry if the mask_obj entry does not exist.

Changes to mask_obj entry are useful for temporary changes. Specify only the permission bits you really want users and groups to have. And use mask_obj entry to temporary restrict some permissions.

6.3.4 Using AIX commands on DFS LFS Files That Have ACLs

AIX commands do not know about DFS LFS ACLs. In cases where you use the AIX commands to move (mv) and copy (cp) files, your ACLs may not be preserved. They work as follows:

If you move an DCE LFS file within the *same* fileset, the DFS LFS ACLs will be preserved. If you move an DCE LFS file to another DCE LFS fileset, the ACL assigned to the new file will be the IOC value from the new parent directory in the new fileset (following the rules that were explained in 6.2.4, “ACL Inheritance” on page 172). In other words, your old ACL information will be removed in favor of the default values for new objects in the new directory.

If you use the AIX cp **command to copy files the ACLs will not be preserved** no matter where you move these files. If you just think that AIX file manipulation does not know about ACLs, it is easy to remember that the ACL information will be lost. If you copy a DCE LFS file to a DCE LFS fileset, the ACL assigned to the new file will be the IOC value from the new parent’s directory following the rules that were explained in 6.2.4, “ACL Inheritance” on page 172.

6.3.5 Effects of Showing the Wrong File Ownership

It is possible to incorrectly list the proper owner of a file or a directory. This happens when the UIDs for users on a machine do not have the same DCE user id in the security registry. For example, if takeda is an AIX user and his UID is 200, then his UID for his DCE login should be the same. This does not happen automatically and care must be taken to set these to match. See the passwd_export and passwd_import DCE commands for information on how to make the DCE user IDs match the AIX user IDs. The same is also true for the DCE defined groups and the AIX defined groups. We will now show an example of how confusing this can be if the AIX UIDs do not match the DCE UIDs.

When a file or directory is created in DFS, the username and groupname represent user IDs (UIDs) and group IDs (GIDs) that are stored in the DCE security registry. When AIX commands like ls -l are used, the AIX commands try to interpret the UIDs and GIDs that belong to the file. For example:

```
# cd /:/dir1
# li -l
Frw-rw-r--    1 guest    12             0  Nov 15 12:16 file1
```

In this case, file1 appears to be owned by guest and the 12 group. Let us look at the /etc/passwd file.

```
# cat /etc/passwd
root:!:0:0::/:/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/usr/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/usr/guest:    <--------
nobody:!:4294967294:4294967294::/:
lpd:!:104:9::/:
ron:!:200:1::/u/ron:/bin/ksh
dfsrb:!:201:1::/u/dfsrb:/bin/ksh
```

We can see from this that the UID associated with guest is 100. AIX thinks that 100 represents the user guest. If we look at the information in the security registry, we see the following:

```
rgy_edit=> do p
Domain changed to: principal
rgy_edit=> v
nobody                                          -2
root                                             0
daemon                                           1
sys                                              2
bin                                              3
uucp                                             4
who                                              5
mail                                             6
tcb                                              9
dce-ptgt                                        20
dce-rgy                                         21
cell_admin                                     100 <----------------
krbtgt/saurus                                  101
hosts/sys4/self                                102
hosts/sys4/cds-server                          103
hosts/sys4/dfs-server                          106
ron                                            107
takeda                                         108
junk                                           109
```

The real owner of this file is cell_admin because he is the principal with a UID that is 100.

The same can be shown with the group. In this case, AIX cannot resolve the 12 group to a name. If we look in /etc/groups, this is what we see:

```
# cat /etc/group
system:!:0:root
staff:!:1:dfsrb,ron
bin:!:2:root,bin
sys:!:3:root,bin,sys
adm:!:4:bin,adm
uucp:!:5:uucp
mail:!:6:
security:!:7:root
cron:!:8:root
printq:!:9:lpd
```

```
audit:!:10:root
ecs:!:28:
nobody:!:4294967294:nobody
usr:!:100:guest
```

Note that a group ID of 12 is not defined. That is why AIX could not resolve it to a name. If we look at the group domain in rgy_edit, we can see the true group owner.

```
rgy_edit=> do g
Domain changed to: group
rgy_edit=> v
nogroup                                    -2
system                                      0
daemon                                      1
uucp                                        2
bin                                         3
kmem                                        4
mail                                        6
tty                                         7
none                                       12  <-----------------
tcb                                        18
acct-admin                                100
subsys/dce/sec-admin                      101
subsys/dce/cds-admin                      102
subsys/dce/dfs-admin                      103
subsys/dce/dts-admin                      104
subsys/dce/dskl-admin                     105
subsys/dce/cds-server                     106
subsys/dce/dts-servers                    107
subsys/dce/dfs-fs-servers                 108
subsys/dce/dfs-bak-servers                109
aaa                                       110
```

The true group owner of file1 is none.

This can get quite confusing if the AIX UIDs and GIDs are not the same as the UIDs and GIDs that are defined in the security registry. For example, we now change the owner on file1 as follows:

```
# chown ron file1
ls -l file1
-rw-rw-r--   1 ron      12             0 Nov 15 12:16 file1
```

Now that we have changed the owner of this file to the AIX user ron, we can note that the DCE user called ron cannot even modify the ACL--even though he appears to own the file.

```
# dce_login ron -dce-
# acl_edit /:/dir1/file1
sec_acl_edit> l

# SEC_ACL for /:/dir1/file1:
# Default cell = /.../saurus
mask_obj:rw----
user_obj:rw-c--
group_obj:r-----
other_obj:r-----
sec_acl_edit> m user:takeda:rw----
sec_acl_edit> co
```

```
ERROR: operation on acl not authorized (dce / sec)
sec_acl_edit>
```

This is because the user ron has a UID of 200 from the /etc/passwd file. There is not a user defined as 200 in the security registry. Therefore, the only ones that can now modify this ACL are the members of the dfs-admin group. (The cell_admin is a member of this group by default.)

```
rgy_edit=> v subsys/dce/dfs-admin -m
subsys/dce/dfs-admin                          103
  2 members
    cell_admin, hosts/sys4/dfs-server
```

Always make sure DCE UIDs Match AIX UIDs

Much of this confusion does not exist if care is taken so that whenever you define AIX users that their UID matches their DCE login ID. All groups defined in AIX should also have corresponding DCE group IDs. See the passwd_import and passwd_export commands in IBM InfoExplorer to help correlate the AIX machines with the security registry.

6.3.6 Effects of foreign_users Storing Files In Your DFS Filespace

If your cell's DFS file space is set up to allow write access by unauthenticated users (any_other ACL entries) or users from foreign cells (foreign_user, foreign_group, or foreign_other ACL entries), the default cell associated with a file or directory they create is determined by the cell of the user creating the object not by the local cell of the DFS file space where the object is created. The default cell associated with the object affects evaluation of the ACL entries.

If a foreign user created the object, then the default cell is the foreign user's cell. If an unauthenticated user created the object, the default cell is listed as unknown. The user_obj, group_obj, other_obj and all user and group entries are evaluated relative to the default cell.

For example, user ron from cell /.../saurus creates the file, ronfile, in the dino cell's DFS file space. The file is owned by the user /.../saurus/ron and the user_obj, group_obj, other_obj and all user and group entries are evaluated relative to the /.../saurus cell not the /.../dino cell. The ACL would look like this:

```
$ acl_edit ronfile -l

# SEC_ACL for ronfile:
# Default cell = /.../saurus
mask_obj:rw----
user_obj:rw-c--
group_obj:rw----
other_obj:rw----
```

With the above ACL, only users in the /.../saurus cell are given access to the file. Users in the /.../dino cell would not have access to the file. They would would have to be given access using the foreign_user, foreign_group, foreign_other, or any_other ACL entries.

If an unauthenticated creates a file in the dino cell's DFS file space, the file is owned by the user nobody and the cell is unknown. The ACL would look like this:

```
$ acl_edit newfile -l
Unknown default cell from ACL - ERROR: Cell UUID is not a valid cell name (dce /
 sec)
 INFO: Local cell will be used for operations requiring default cell info.

 # SEC_ACL for newfile:
 # Default cell = ffffffffe0000.00.00.00.00.00.00.00.00
 mask_obj:rw----
 user_obj:rw-c--
 group_obj:rw----
 other_obj:rw----
```

The error displayed by the acl_edit command stating that the cell UUID is not a valid cell name is normal and can be ignored.

With the above ACL, only unauthenticated users would be able to gain access to this file. Users from the /.../dino cell or any foreign cell would have to be given access using the foreign_user, foreign_group, foreign_other, or any_other ACL entries.

In general, you probably will not allow unauthenticated users to write in the DFS file space but you might want to allow them read access to public directories. If you are in an Intercell environment, you may need to allow foreign users write access to areas your DFS file space.

The following example shows how part of the /.../dino cell has been set up as a repository for public, non-restricted information available to all local cell users, foreign cell users, and unauthenticated users. The directory's ACL is set up with other_obj, foreign_other, and any_other entries allowing read and execute (search) permissions. The directory's initial container ACL is set up with other_obj and foreign_other entries allowing read, write, execute (search), insert, and delete permissions. The directory's initial object ACL is set up with other_obj and foreign_other entries allowing read, write, execute permissions. It is important to get the initial object and initial container ACLs set properly to guarantee access to the files and directories created below the public directory.

The owner of the /.../dino/fs/public directory is set up to be cell_admin and the group is dfs-admin.

The ACL for /.../dino/fs/public is set up to look like this:

```
$ acl_edit /.../dino/fs/public -l
# SEC_ACL for /.../dino/fs/public:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
group_obj:rwx-id
other_obj:rwx-id
foreign_other:rwx-id
any_other:r-x---
```

The initial container ACL for /.../dino/fs/public is set up to look like this:

```
$ acl_edit /.../dino/fs/public -ic -l
# Initial SEC_ACL for directories created under: /.../dino/fs/public:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
group_obj:rwx-id
```

```
other_obj:rwx-id
foreign_other:rwx-id
any_other:r-x---
```

The initial object ACL for /.../dino/fs/public is set up to look like this:

```
$ acl_edit /.../dino/fs/public -io -l
# Initial SEC_ACL for directories created under: /.../dino/fs/public:
# Default cell = /.../dino
mask_obj:rwx---
user_obj:rwxc--
group_obj:rwx---
other_obj:rwx---
foreign_other:rwx---
any_other:r-x---
```

6.4 Foreign Cell Setup

It is important to understand how ACLs work in an multi-cell environment. The following sections present how to set up this type of environment and how the ACLs can be used to protect DCE LFS files and directories.

6.4.1 Multi-Cell Environment

Administering a Multi-Cell Environment requires additional tasks and considerations due to the interaction of principals across different cells. If you decide to set up groups for two types of administrators, you can set the ACL for the *krbtgt* directory in the registry database to allow updating only by the group of intercell administrators. Doing that you are going to define all others users read access to this directory and/or intercell access will be denied to those users.

In addition, the DFS ACL entries set permissions for:

1. Specific principals in the local and foreign cells
2. Specific groups in the local and foreign cells
3. All other principals in a specific foreign cell for whom individual permissions have not been set
4. All principals in any cell who have been authenticated by the DCE Authentication Service

To give explicit permission for principals, we have to define a special account in the local cell's registry to represent the foreign cell and a special account in the foreign cell registry to represent your cell. See Appendix A, "DCE Intercell Communication Setup and ACLs" on page 215 section A.3.6, "Create the Cross-Cell Authentication Accounts" on page 220.

6.4.2 Setting Up ACLs for Foreign Cells

The default ACLs for the root DFS file system in a cell is as follows:

```
acl_edit /.../sauro/fs
sec_acl_edit> l

# SEC_ACL for /.../sauro/fs:
# Default cell = /.../sauro
user_obj:rwxcid
```

```
group_obj:rwx-id
other_obj:rwx-id
```

If a foreign user is to be allowed to access to files and directories in this cell, you must add an any_other type entry in the local cell at least with rx permission. For example, the following will add this entry:

```
sys2>acl_edit /.../dino/fs
sec_acl_edit> l

# SEC_ACL for /.../dino/fs:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
group_obj:rwx-id
other_obj:rwx-id
sec_acl_edit> m any_other:rwx-id
sec_acl_edit>
```

This allows users from other cells to read and list the contents of directories as well to read and execute files. To understand the access checking algorithm in the intercell environment we will present the following examples.

Consider the following principals and groups included in the dino cell and in the sauro cell.

```
Cell name: dino
rgy_edit=> do acc
rgy_edit=> v
aws [none ibm]:*:212:12::/home/root/aws:ksh:
ps3 [software ps]:*:117:113::/::
u-dino [dino ibm]:*:210:212::/::

Cell name : sauro
rgy_edit=> do acc
rgy_edit=> v
sauro1 [oem aws]:*:117:113::/home/root/aws:ksh:
sauro2 [software aws]:*:118:112::/home/root/aws:ksh:
sauro3 [oem aws]:*:119:113::./home/root/aws::
u-sauro  [sauro ibm]:*:205:213::/::
```

As an example, sauro1 is a member of the oem group. The UID for sauro1 is 117. The GID for the oem group is 113. The members of the oem group (GID=113) are sauro1 and sauro3.

6.4.2.1 Example 1: Allowing a Foreign User Access To a File

This example show that a foreign user can access a file in a local file system only if he has read and execute permission for that local file system.

The following shows a directory /.../dino/fs/testlfs-1 with these permissions:

```
# SEC_ACL for /.../dino/fs/testlfs-1:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
user:aws:rwx-id
foreign_user:/.../sauro/sauro2:rwx-id
group_obj:r-x---
```

```
foreign_group:/.../sauro/oem:rwx-id
other_obj:r-x---
```

The permission to read, write and execute will granted to the users explicitly declared under this ACL. They are: aws, sauro2 and all users that belong to the foreign group called oem. The users in the /.../sauro/oem group are sauro1 and sauro3.

If you try to cd to /.../dino/fs/testlfs-1 after doing a `dce_login` as /.../sauro/sauro2 and it returns a permission denied message, verify the /.../dino/fs permission set. You will need to add permission for members of a foreign cell to access objects in this local cell's file system.

This is done by adding `sec_acl_edit> m any_other:rx` to the fs ACL. Foreign users are then allowed to list the files and directories in the local cells file system.

6.4.2.2 Example 2: The Default Cell and File Ownership

To determine file ownership, you must know the default cell.

The following shows the /.../dino/fs/testlfs-2 directory with the following permissions:

```
sys2:/.../dino/fs/testlfs-2>acl_edit testlfs-2 -l

# SEC_ACL for /.../dino/fs/testlfs-2:
# Default cell = /.../dino
mask_obj:rwx-id
user_obj:rwxcid
user:ps3:rwx-id
foreign_user:/.../sauro/sauro1:rwx-id
group_obj:r-x---
other_obj:r-x---
sec_acl_edit>
```

Looking at the previous registry databases:

```
Cell name: dino
rgy_edit=> do acc
rgy_edit=> v
ps3 [software ps]:*:117:113::/::

Cell name : sauro
rgy_edit=> do acc
rgy_edit=> v
sauro1 [oem aws]:*:117:113::/home/root/aws:ksh:
```

Here is a case where the local user has the same UID as the foreign user. To make it even more confusing, the local user could also have the same GID as the foreign user.

If we list the directory:

```
sys2:/.../dino/fs/testlfs-2>ls -l
-rw-r--r--   1 117      113            0 Jan 12 17:07 dino-test
drwxr-xr-x   2 117      113          256 Jan 12 15:29 ps3-dir1
drwxr-xr-x   2 117      113          256 Jan 12 15:30 dir2
-rw-r--r--   1 117      113            0 Jan 12 17:10 sauro-test
```

We see that dino-test, ps3-dir1, dir2 and sauro-test all appear to be owned by UID=117 and GID=113. UID=117 is the ps3 user on the dino local cell. This file may also belong to the sauro1 user on the sauro (foreign) cell. A similar situation occurs for the groups. A GID=113 can either be the software group on the dino cell or the oem group on the sauro cell.

This shows that you cannot tell who owns a file or a directory by using an `ls -l` command unless you know what the default cell is. The default cell is defined in the ACL for the object.

6.4.2.3 How To Determine File Ownership

The /.../sauro/fs/user1 directory is created and given the following permissions:

```
# SEC_ACL for user1:
# Default cell = /.../sauro
mask_obj:rwx-id
user_obj:rwxcid
foreign_user:/.../dino/u-dino:rwx-id
group_obj:r-x---
other_obj:r-x---
sec_acl_edit> e
```

The foreign_user:/.../dino/u-dino has the permission to write into this directory. The authenticated user u-dino now creates a directory in the /.../sauro/fs/user1 directory called ddir. It looks like as follows:

acl_edit /.../sauro/fs/user1/ddir -l

```
# SEC_ACL for /.../sauro/fs/user1/ddir:
# Default cell = /.../dino
user_obj:rwxcid
group_obj:r-x---
other_obj:r-x---
sec_acl_edit>
```

Note the default cell is dino, even though it is stored in the sauro cell. It however, is owned by a user from the /.../dino cell.

The authenticated user u-sauro can now create a file in the same directory.

```
dce_login u-sauro <type passwd>
cd /.../sauro/user1
mkdir sdir
```

Now, authenticate as /.../dino/u-dino, and list this directory from the dino cell:

```
dce_login u-dino
cd /.../sauro/fs/user1
ls -l
drwxr-xr-x   2 u-user2  ibm          256 Jan 14 11:46 ddir
drwxr-xr-x   2 u-user3  213          256 Jan 14 11:42 sdir
```

If you are authenticated as u-sauro from the local cell, the same directory /.../sauro/fs/user1 now appears to have different owners.

Listing the file as u-sauro from the sauro cell:

```
dce_login u-sauro
cd /.../sauro/fs/user1
ls -l
drwxr-xr-x   2 210      212          256 Jan 14 11:46 ddir
drwxr-xr-x   2 u-user1  ps           256 Jan 14 11:42 sdir
```

Why do they have different owners and groups? How can you figure out who is the correct owner and group of ddir and sdir directories? It is done as follows:

1. From the dino cell, the directory looks like this:

   ```
   drwxr-xr-x   2 210      212          256 Jan 14 11:46 ddir
   drwxr-xr-x   2 u-user1  ps           256 Jan 14 11:42 sdir
   ```

   When you issue the AIX command :

   - **id u-user2**.

     The system shows that there is an AIX UID for AIX user u-user2.
     uid=210(u-user2) gid=212(ibm)

   - **id u-user3**.

     The system shows that there is an AIX UID for AIX user u-user3.
     uid=205(u-user3) gid=1(staff)

     When you issue the DCE command:

   - rgy_edit -v |grep 210

     The system answer is:

     u-dino [dino none]:*:210:212::/:

     Issuing the same command to uid=205, you do not receive an answer. This means that there is not any principal using UID=205.

2. From the sauro cell, the directory looks like this:

   ```
   drwxr-xr-x   2 210      212          256 Jan 14 11:46 ddir
   drwxr-xr-x   2 u-user1  ps           256 Jan 14 11:42 sdir
   ```

   When you issue the AIX command :

   - **id 210**

     The system shows that there is not an AIX UID for AIX user **210**.
     3004-820 User not found in /etc/passwd file

   - **id u-user1**

     The system shows that there is an AIX UID for AIX user **u-user1**.
     uid=205(u-user1) gid=213(ps)

     When you issue the DCE command:

   - rgy_edit -v |grep 205

     u-sauro [sauro ibm]:*:205:213::/::

The following table summarizes the information we listed above:

System	Belongs to cell	Verifying local AIX user on each system	Verifying Account Name	Verifying Principal UID	Verifying Group GID
sys2	dino	uid=210(u-user2) gid=212(ibm) --- uid=205(u-user3) gid=1(staff)	u-dino	210	212
sys4	sauro	uid=205(u-user1) gid=213(ps) --- 210=user not found	u-sauro	205	213

Default ACL for each file:

directory	Default ACL
ddir	cell /.../dino
sdir	cell /.../sauro

Directory ddir has #Default cell = /.../dino from its ACL. Looking at sys2 on dino's cell you will find: ddir was created by AIX user u-user2 and DCE authenticated user u-dino.

Directory sdir has #Default cell = /.../sauro from its ACL. Looking at sys4 on sauro's cell you will find: sdir was created by AIX user u-user1 and DCE authenticated user u-sauro.

Note

When doing a `ls -l`, the owner and group name will come from the local /etc/passwd and /etc/group files if AIX can match the UID + GID number. If AIX cannot match a file UID and/or GID to a name, they will be the uid and gid number of the principal and group in DCE registry database.

The flowchart in Figure 55 on page 194 shows how to determine the ownership of files and directories.

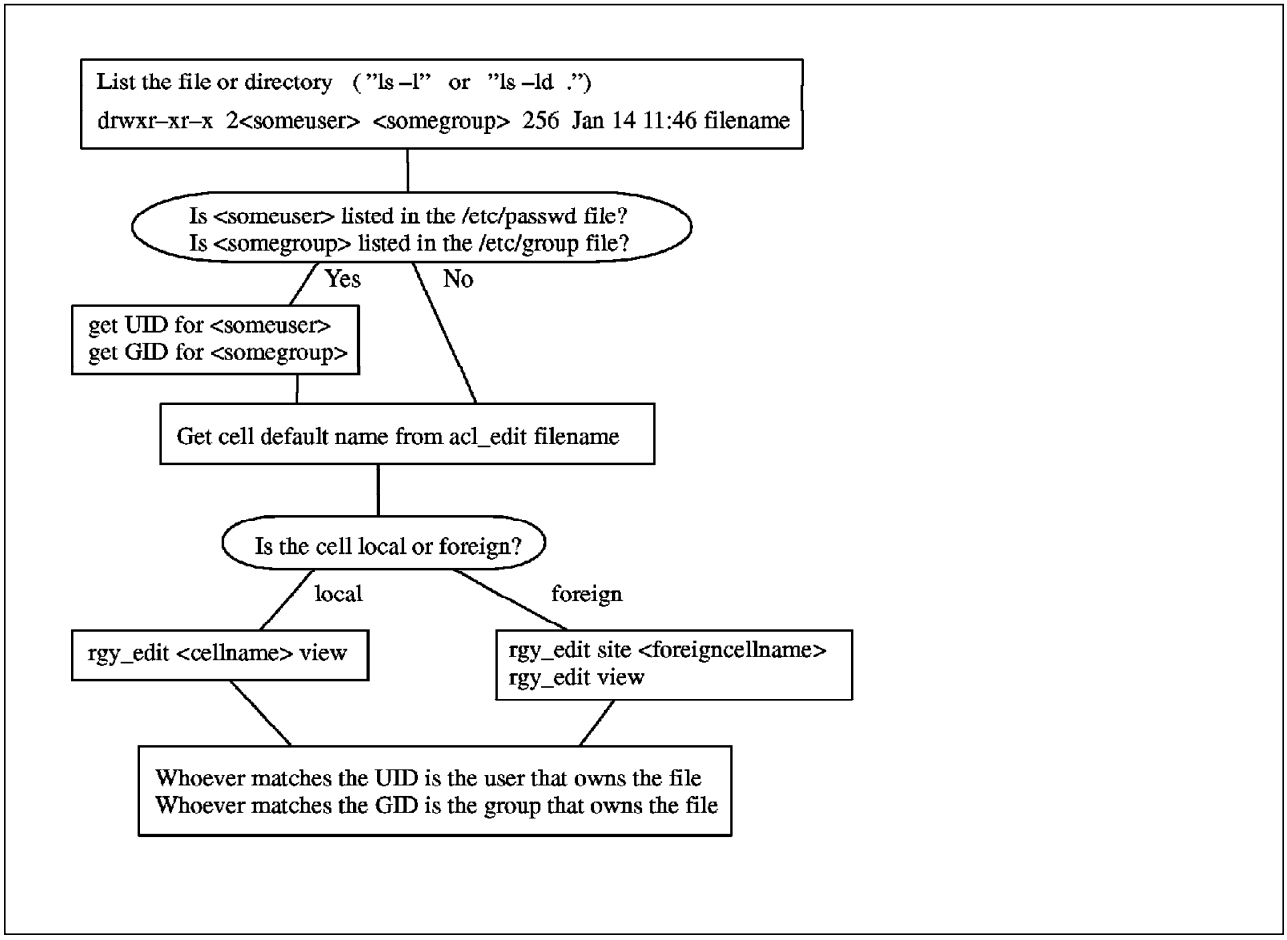


Figure 55. Algorithm of How To Determine The Ownership of a File or Directory

Chapter 7. Some Real Life Examples with DFS

The Distributed File System (DFS) can be very helpful when making large amounts of data available to many users. It is even more efficient if the data is read-only because it simplifies the management of the cache on DFS clients. Areas which are likely to be used with DFS include distribution of installation images and PTFs (program temporary fixes) or just executables which can be used from many sites without being duplicated. Using IBM InfoExplorer is very efficient with DFS. It eases the maintenance and saves plenty of local storage resources so they can be used for other purposes.

This chapter describes the use of DFS as a:

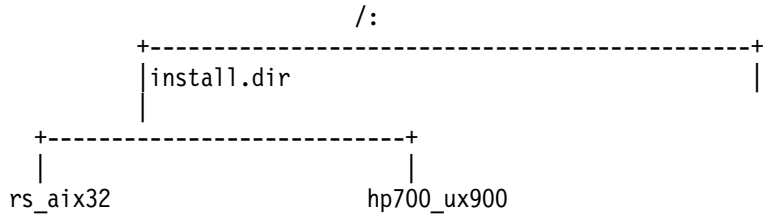
1. Common Installation image distributor
2. Common documentation distributor (IBM InfoExplorer with DFS)
3. Common binary file distributor (distribution of executables)

The basic method in all three examples is very similar. Files and directories are distributed via DFS.

7.1 DFS as a Common Installation Image Distributor

The installation images for AIX Version 3 and its LPPs are a good example of data that could be distributed with DFS. They are big, have a nature of being read-only and need to be shared amoung many users. The major advantage is that they have to be placed on the file system only once thus saving space. It also is much faster installing from DFS than installing from a tape. Every member in the DCE DFS environment has access to the installation images and the binaries only need to be maintained in one place.

If you have several versions of the AIX operating system or other vendor's operating system to maintain, we suggest you put the installation images of different versions into separate directories. For example, put the installation images for AIX Version 3.2 in /:/install.dir/rs_aix32, and the images for HP Ultrix** in /:/install.dir/hp700_ux900 such as outlined below:



Furthermore, we want the installation images to be seen on local systems always under the path name /usr/sys/inst.images. This can be achieved by using symbolic links with the @sys variable. The @sys variable is set by default on different operating systems DFS implementations. For example, on AIX Version 3.2 it has the value "rs_aix32". The variable can be queried with the `cm sysname` command and can be set with `cm sysname -newsys <name>` command. The directory name that you use for storing the install images should match the default name for a given platform.

The following steps outline the procedure to set up such an installation server environment with DFS.

- Step 1: Create the fileset install.dir

  You will need a fairly large fileset to hold all installation images for several operating systems. In our example, we create on one of the File Server Machines (/.:/hosts/sys3) a fileset with 200MB (this is equivalent to 50 physical partitions each having 4MB) on the root volume group (rootvg). If you need more information on how to create filesets, refer to 5.3.1, “Creating DFS Aggregates and Filesets” on page 114.

  ```
  # mklv -y 'lfsinstall' rootvg 50
  # newaggr -aggregate lfsinstall -bl 8192 -fr 1024 -overwrite
  # mkdfslfs -f install -m /:/install.dir -d /dev/lfsinstall -n lfsinstall
  ```

  You will need the appropriate permissions on the file server and on the FLDB in order to do the above commands. At this time, you may set the ACLs for the /:/install.dir mount point as well as for the initial object creation (-io) and initial container creation (ic) objects. We will mount the fileset directly to the root.dfs, but this could be any place in the hierarchy.

- Step 2: Copy the images into the appropriate directories

  Before we place the images into the DFS filespace, we create several directories; one for each operating system version that we want to support.

  ```
  # mkdir /:/install.dir/rs_aix32
  # mkdir /:/install.dir/rs_aix33
  # mkdir /:/install.dir/hp_ultrix
  ```

  Then use any method available to you to place the images into the appropriate directories. For example, use SMIT to copy the images from an installation tape to the correct directory.

  After placing the images into the appropriate directories, you have to plan on who should have access to these directories. You would then use

  ```
  acl_edit
  ```

  on the directories and files to provide the correct access rights. For example:

  ```
  # cd  /:/install.dir/rs_aix32
  # find . -exec chgrp adm {} \;
  # find . -exec chown adm {} \;
  # find . -exec acl_edit {} -m other_obj:r-x--- \;
  # find . -exec acl_edit {} -m group_obj:rwxcid \;
  ```

  This would provide read and execute access for all users so that they can install the images and it would provide all access rights to members of the group adm to maintain the directory. This is just an example and you may find other solutions more appropriate for your environment.

- Step 3: Establish symbolic links on each DFS client machine

  The last step is to provide the correct links on each of the DFS client systems to reach the installation images from the correct path.

  ```
  # ln -s /:/install.dir/@sys  /usr/sys/inst.images
  ```

  The installation images are now available from each system under the path name /usr/sys/inst.images. The link statement is the same on each DFS

client system and can therefore be integrated in shell scripts such as profiles for use at system restart.

7.2 DFS as a Common Documentation Distributor

IBM InfoExplorer is a good example of software that could be distributed via DFS. The product consists of a few configuration files and a number of fairly large databases. With DFS the databases can be distributed to all sites but only need to be stored physically on one site. This will save a lot of space on the majority of DFS clients. This is a very useful method for system administrators to reduce the effort it takes to update these databases since they are kept only on one server system.

This method is very similar to using the IBM InfoExplorer databases from a CD-ROM except you don't link to the CD-ROM. Instead you link to the DFS filespace. We will demonstrate how to set this up using the US English national language support. Note that the procedure for other national languages is different only in the way that you use another name instead of En_US for the language to be installed. The main steps involved in setting up IBM InfoExplorer in US English on DFS are as follows:

- Step 1: Install IBM InfoExplorer basic files on each DFS client
  - The installation of IBM InfoExplorer involves only the basic files. The databases are separately installable options which come with each LPP. Installation of IBM InfoExplorer is not difficult. Use ″SMIT install″ and install the image bos.data if it is not already installed on your DFS client. Check for this with the following command:

    ```
    # lslpp -L bos.data
    ```

    If you want to run IBM InfoExplorer from an X-Windows screen, then the AIXwindows Runtime Environment must also be installed.
  - Change to the directory /usr/lpp/info and delete anything which is stored under the directory En_US (the language switching variable). In this case, En_US will be our link point to the databases coming from the DFS filespace.

    ```
    # cd /usr/lpp/info
    # rm -r /usr/lpp/info/En_US
    ```

    Note

    If this system will be the File Server for the IBM InfoExplorer databases, then do not delete them now. Wait until you have copied them into the DFS namespace before deleting them.
  - Change to the /usr/lpp/info/data directory and edit the ispaths file. List all databases to which you want access. You may want to copy the file ispaths.full to ispaths. This would provide you with all databases. The following is just an example, there are more databases.

    ```
    # @(#)13        1.7.1.1  R2/cmd/pubs/ispaths.full, cmdpubs, bos320,....
    # COMPONENT_NAME: (cmdpubs) ispaths.full
    #
    # FUNCTIONS: n/a
    #
    # ORIGINS: 43
    #
    ```

```
# (C) COPYRIGHT International Business Machines Corp. 1990,1991
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# (Copyright statements and/or associated legends of other
# companies whose code appears in any part of this module must
# be copied here.)

################################################
#         info Navigation Database             #
################################################

id              0
primnav         TRUE
help            TRUE
name            nav
title           Navigation
sys             /usr/lpp/info/%L/sys.sys
key             /usr/lpp/info/%L/nav/nav.key
rom             /usr/lpp/info/%L/nav/nav.rom

################################################
#         info Using & Managing Database       #
################################################

id              1
glossary        TRUE
name            aix
title           Using and Managing
sys             /usr/lpp/info/%L/sys.sys
key             /usr/lpp/info/%L/aix/aix.key
rom             /usr/lpp/info/%L/aix/aix.rom

################################################
#         info Programming Database            #
################################################

id              2
name            prog
title           Programming Base OS, Commun, Devices
sys             /usr/lpp/info/%L/sys.sys
key             /usr/lpp/info/%L/prog/prog.key
rom             /usr/lpp/info/%L/prog/prog.rom


################################################
#               info Update Database           #
################################################

id              3
name            update
title           Interim Updates to Documentation
sys             /usr/lpp/info/%L/sys.sys
key             /usr/lpp/info/data/%L/update/update.key
rom             /usr/lpp/info/data/%L/update/update.rom
```

```
##############################################
#               info dce Database            #
##############################################

id              4
name            dce
title           DCE Programming
sys             /usr/lpp/info/%L/sys.sys
key             /usr/lpp/info/%L/dce/dce.key
rom             /usr/lpp/info/%L/dce/dce.rom

##############################################
#               info Encina Database           #
##############################################

id              5
name            encina
title           Encina Transaction Processing Environment
sys             /usr/lpp/info/%L/sys.sys
key             /usr/lpp/info/%L/encina/encina.key
rom             /usr/lpp/info/%L/encina/encina.rom
```

- Step 2: Install the IBM InfoExplorer databases on a DFS File Server
  - On one of the File Servers which has enough space to hold all the IBM InfoExplorer databases (can be several hundred megabytes), we must create a fileset and mount it into the DFS namespace. Any mount point is acceptable and for simplicity we have chosen /:/info.

    ```
    # mklv -y 'lfsinfo' rootvg 50
    # newaggr -aggregate lfsinfo -bl 8192 -fr 1024 -overwrite
    # mkdfslfs -f info -m /:/info -d /dev/lfsinfo -n lfsinfo
    ```

  - Set up the ACLs for the /:/info mount point, the initial object creation object (`acl_edit /:/info -io`) and the initial container object (`acl_edit /:/info -ic`) at this time.
  - Create a directory on the File Server with the name En_US and copy all available databases into this directory.

    ```
    # cd /:/info
    # mkdir En_US
    # cp -rp /usr/lpp/info/En_US/*  /:/info/En_US
    ```

- Step 3: From each of the DFS clients which you want to allow IBM InfoExplorer access now create a symbolic link to the appropriate databases.

  ```
  # cd /usr/lpp/info
  # ln -s /:/info/En_US  /usr/lpp/info/En_US
  ```

To Use InfoExplorer From Within X-Windows

Before IBM InfoExplorer can be used from within X-Windows, you need to restart the infod daemon after you have logged into DCE. This is because the daemon must run with the credentials of a DCE user. Use kill -9 <PID> and restart the infod daemon in the background.

You can now use IBM InfoExplorer from the DFS client machines by entering `info` at the command line.

7.3 DFS as a Binary File Distributor

In this example, we will describe how to make binaries installed in the DFS filespace available to DFS client systems thus reducing the amount of storage and maintenance for these binaries.

Useful programs to be placed into this directory are AIXwindows examples or any other general purpose executables. Binaries for a specific machine type, such as the RISC/6000 running AIX 3.2 can only be used on DFS client machines that are RISC/6000′s running AIX 3.2. To keep the same pathname from all DFS client machines regardless of the operating system type, we will use the @sys feature when setting up the binaries in the DFS filespace. For more information about the @sys variable, see the ″Using the @sys and @host Variables″ section in the DFS Configuration Issues chapter of the *DCE Administration Guide*. This example only shows adding AIX 3.2 RISC/6000 binaries. If you are supporting a cell with DFS clients running on different operating system/hardware types, you will want to add similar binaries for each of the platforms following the steps below.

To run these steps you should be locally logged in as the root id on the DFS File Server and have done a dce_login as a DCE principal with the proper authority to create the fileset and rwxid permission in the DFS root dir to mount the fileset (usually the DCE root principal or the cell_admin principal).

The main steps are:

1. Create and export a new aggregate on a DFS File Server System. If you have an existing aggregate with space available you can skip this step.

   ```
   # mklv -y lfsbin -t lfs rootvg 20
   # newaggr -aggregate lfsbin -bl 8192 -fr 1024
   # mkdfslfs -d /dev/lfsbin -n lfsbin
   ```

2. Create a fileset rs_aix32.bin and mount it at /:/rs_aix32/bin

   ```
   # mkdir /:/rs_aix32
   # mkdfslfs -f rs_aix32.bin -m /:/rs_aix32/bin -n lfsbin
   ```

3. Use the `acl_edit` command to check the /:/rs_aix32/bin directory′s ACLs including the initial container creation acl (ICC) and the initial object creation acl (IOC).

   ```
   # acl_edit /:/rs_aix32/bin -l
   # acl_edit /:/rs_aix32/bin -ic -l
   # acl_edit /:/rs_aix32/bin -io -l
   ```

   This will show you the default settings for all three ACLs. You can then use the `acl_edit` command to modify the acls to fit your needs.

4. Install the binaries in the /:/rs_aix32/bin directory. Any method available to you such as `cp`, `tar` or `rcp` is acceptable.

5. Using the @sys variable, provide a symbolic link in the DFS filespace from /:/local/bin to ../@sys/bin so that all DFS users can access the binaries using the pathname /:/local/bin regardless of the OS type of their DFS client machine. (When the DFS client encounters @sys in a path name, it replaces

the variable with a defined system name. Use the cm sysname command to determine the @sys value for a particular DFS client machine).

```
# mkdir /:/local
# ln -s ../@sys/bin /:/local/bin
```

Users at any DFS client machine can now execute those binaries by using the DFS path /:/local/bin/<binary_name>.

Users can also add the DFS path name, /.../<cell_name>/fs/local/bin to their PATH environment variable or they can set up a symbolic link from /usr/local/bin to /.../<cell_name>/fs/local/bin. When adding a DFS path name to your PATH environment variable, use the fully qualified path name beginning with /.../<cell_name>/fs instead of /: because the ″:″ will look like a separator character to most shells such as /bin/ksh.

To put the DFS pathname in your PATH environment variable:

```
# PATH=$PATH:/.../dino/local/bin::
# export $PATH
```

To set up a symbolic link from /usr/local/bin to the DFS pathname:

```
# ln -s /.../dino/fs/local/bin /local/bin
```

Chapter 8. DFS Troubleshooting

The examples given in this chapter are useful for the administrator that may need to do some DCE and/or DFS troubleshooting.

8.1 DFS Configuration Failures

To configure DFS you need to run the `mkdfs` shell script in the following manner depending on which machine role you are setting up:

- `mkdfs dfs_scm` (configures the system control machine)
- `mkdfs dfs_fldb` (configures the fileset location database)
- `mkdfs dfs_svr` (configures a DFS file server)
- `mkdfs dfs_bkdb` (configures the backup database)
- `mkdfs dfs_cl` (configures a DFS client)

This shell script is located in the /usr/lpp/dce/bin directory. In some cases the shell script returns with error messages which leave it unclear to the administrator how to solve the problem. It is recommended that you always run the `rmdfs` shell script to clean up the failed DFS installation. The `rmdfs` will eventually clean up some elements which may have been left configured even if the `mkdfs` failed. You may also view the `mkdfs` shell script and search for the error message.

8.2 General Guidelines for Debugging DFS Problems

DFS provides access to a distributed file system. If you are having problems accessing files, directories, file servers or other DFS-related items, the following may be used to assist you in diagnosing your DFS access problem.

8.2.1 Is the Network Operational?

Check if your local workstation has the network environment working. Try a `ping` to some machine on your network. This will test whether you have a hardware problem with your network card (usually token ring or ethernet) or its connections. This will also test to make sure that TCP/IP is running properly. You must have TCP/IP running to run DCE and DFS.

8.3 Is DFS Functioning Properly?

1. Check if you have the DFS client on your own machine:

   # df

   It should have one standard line as:

   ```
   DFS              9000000  9000000     0%        0      0% /...
   ```

   If not, the DFS client process is not active on your machine. Run /etc/rc.dfs. You need root authority to do this.

2. Check if the DFS file servers are running by issuing the following command:

   # cm statservers

   The usual answer is: *All servers are running*.

a. If you get: *These servers are still down: sys1, sys2,* . It means that File Server machines sys1 and sys2 did not respond to this call and the DFS Cache Manager marks them as being down. These machines may be experiencing a DCE/DFS failure or a hardware failure.

b. If you get: *cm: Function not implemented*, this means that there are no DFS processes running on this machine.

3. Try to cd to the DFS filespace:

   # cd /:

   This should change you to the DFS root directory for the cell. If the answer to cd /.:/fs is *ksh: fs:permission denied*, this means that you do not have the proper authority to access the root of the DFS filespace. Check that you have done a dce_login by issuing the klist command.

4. List the cells contacted since the host was rebooted.

   cm lscellinfo

   You will receive some important information about your cellname and all hostnames that have accessed the cellspace.

   The response is similar to:

   ```
   Cell dino on hosts sys1.dce-dfs.austin.ibm.com, sys2.dce-dfs.austin.ibm.com
   , sys3.dce-dfs.austin.ibm.com.
   ```

5. Check the status of all machines which have been defined with some server role in your cell space:

   bos status -s <hostname>

   a. The standard answer is:

   ```
   root@sys2:/>bos status -s sys1
   bos: WARNING: short name for server used; no authentication information will be
   sent to the bosserver
   Instance upclient.scm, currently running normally.
   Instance flserver, currently running normally.
   ```

   b. Instead, if you have this answer:

   ```
   root@sys2:/>bos status -s sys1
   bos: WARNING: short name for server used; no authentication information will be
   sent to the bosserver
   bos: failed to contact host's bosserver (communications failure (dce / rpc))
   ```

   It means the **bosserver** of this specific machine is not active. Check if DFS is active on this machine. This is OK if this machine is just a DFS client.

8.3.1 Are You an Authenticated or an Unauthenticated user?

1. As an authenticated user

   You must do a dce_login.

   # dce_login <dceloginuserid>

   If you forgot your id, try searching for it using the command:

   # rgy_edit -v

   This command will show the account identification for each user. On the current version of DFS, even though you have a defined home directory the system will not change to this directory automatically when you dce_login.

2. As an unauthenticated user

   If you are an unauthenticated user, you will appear to the system as the user nobody and have permissions as any_other. You may want to verify that the any_other entry type is defined for the DFS root of your cell so that unauthenticated users will have access. To verify it, you can issue:

   ```
   # acl_edit /.:/fs -l
   ```

   This will show you the default cell name and the permissions for the root file directory in your local cell. Of course, you will also need the ″any_other″ entry type in other directories that you wish to access as an unauthenticated user.

8.3.2 Is the DFS Filespace Accessible?

If the DFS filespace is not accessible:

1. Verify the functions of DCE such as Security Service and Cell Directory Services.

   ```
   # dce_login cell_admin -dce-
   ```

2. Check if the File Server Location Database is available and you can list the entries.

   ```
   # fts lsfldb
   ```

3. Check if the File Servers are alive and if they hold valid filesets. Our example shows how to check the fileset headers from sys1, sys2, and sys3.

   ```
   # for H in sys1 sys2 sys3
   # do
   # fts lsheader -server /.:/hosts/$H
   # done
   ```

4. Check if the time is within about a two minute interval (maximum) across all the hosts in your cell.

   ```
   while true
    do
      for H in sys1 sys2 sys3
       do
        echo $H
        rsh $H date
       done
       echo '-------------------------'
       sleep 3
    done
   ```

5. Check if your tickets have not expired and renew them if necessary.

   ```
   # klist
   # kinit
   ```

6. Check to see whether the ACLs caused the problem. You may not have the appropriate access rights. (Note that you may not even have the access rights to list the ACLs.)

   ```
   # acl_edit /:/dir/dir/file -l
   # acl_edit /:/dir/dir/file -m user:cell_admin:rwxcid
   ```

8.3.3 Can You Find the File?

1. Standard `find` command

   The standard UNIX `find` command will work if you change to /.:/fs space. You may not be able to use the `find` command if you use the -user or -group parameter with the `find` command. This is because of the way that AIX tries to interpret the UID and GID numbers as being in the /etc/passwd and the /etc/group files. See the discussion of this in 6.4, “Foreign Cell Setup” on page 188.

2. Would you like to know where the file resides?

   Files are physically distributed in /.:/fs space according to system administrator definition. If you change to /.:/fs space, the command:

   # cm whereis <any_name_in_directory>

   It will help to show which are the name of the cell, the fileset and the host name for this specific file.

   ```
   root@sys1:/.:/fs>cm whereis duser1
   The file 'duser1' resides in the cell 'dino', in fileset 'user1', on host sys3.
   ```

3. Do you have authority to access the file or directory?

   Enter the command:

   acl_edit <filename_or_directoryname> -l

   If you have the proper access rights to list the ACLs, then the principal and/or group id's and permissions will be shown. To find out if you are in one of the groups with access permission, enter the command:

   rgy_edit>klist

   The output from this command will show you your dce_login principal ID and the groups that you are a member of. You can compare that with the ACL on the object to determine what access rights you should have.

4. What operations can you perform on the data?

   Once your name or the name of a group you belong to shows up on an ACL (as shown by the `acl_edit` command), you will see a string of capabilities representing permissions that you have.

5. Is the correct data there?

   If a file's content is suspicious, issue the commands:

   cm flush <filename>

   This command flushes the specified file from your DFS cache.

6. Verify the file system with the DFS cache is not full

   If you are still having a problem accessing your data, verify that the file system containing your DFS cache is not full (AIX `df` command).

8.3.4 Diagnosing Write Problems

When you have trouble writing to files in DFS, try the following.

1. Have you exceeded the fileset quota?

   Run the command:

   fts lsquota <any_file_name>

This will show the name of the fileset, the quota and the % used on the aggregate. When there are 1K free blocks (Quota-Used) and the Partition is less than 100% used, users with the *w* capability can write data.

2. Do you have the correct permission?

   Analyze the ACLs permissions.

3. Are you logged into the correct cell?

   Use the `klist` command to tell you what cell you are logged in to and what user id you have logged in with. Check the cell name. If you are in a foreign cell, check if you have foreign cell identification. The `acl_edit` command will work as:

   acl_edit /.../<cellname>/fs -l

   It will show the permissions for the root DFS file system in the cell that you specify.

   If you are an unauthenticated user in a foreign cell, check for rwx permissions for the any_other entry in the ACL of the directory where you want to write as well as rx permissions on all directories leading to this directory.

4. Is the aggregate full?

   Use the `fts aggrinfo` command to check how much free space is left on the aggregate. If you need more room on the aggregate, see the discussion in 5.3.7, “Increasing the Size of a DCE LFS Aggregate” on page 124 for a discussion of how to do this.

8.3.5 Why Do You See an UUID Format Displayed on a Registry Database?

If a DCE principal or group ID is deleted from the registry database without first deleting objects (files and directories) that it owns in the DFS namespace, its entry in the ACL will be displayed in UUID format. There is no way to delete this entry. You have to adopt it. For example, user5 had access to the /:/duser1 file object. Then the cell_admin deleted the user5 user from the security registry. When the ACL for /:/duser1 is displayed, we see:

```
root@sys1:/.:/fs>acl_edit /:/duser1 -l

# SEC_ACL for /:/duser1:
# Default cell = /.../dino
user_obj:rwxcid
user: 000053e4-64b3-2c75-1005aa881fc:rwx-id
group_obj:rwx-id
other_obj:rwx-id
```

You need to adopt this object as follows:

```
root@sys1:/.:/fs>rgy_edit
Current site is: registry server at /.../dino/subsys/dce/sec/master
rgy_edit=> do p
Domain changed to: principal
rgy_edit=> adopt  000053e4-64b3-2c75-1005aa881f newprinc
```

Now if you display the /:/duser1 object you see:

```
root@sys1:/.:/fs>acl_edit /:/duser1 -l

# SEC_ACL for /:/duser1:
# Default cell = /.../dino
user_obj:rwxcid
user: newprinc:rwx-id
group_obj:rwx-id
other_obj:rwx-id
```

You can now delete the newprinc entry if you want. Note that deleting a user from the security registry can also affect the output of the `lsadmin` command. For example, user5 was a member of the admin.ft administration list until he was deleted. If you now display the members of the admin.ft group, you will see:

```
root@sys1:/.:/fs>bos lsadmin /.:/hosts/sys3 -adminlist admin.ft
Admin Users are: user: hosts/sys3/dfs-server, user: hosts/sys1/dfs-server,
user: 000053e4-64b3-2c75-1005aa881fc, group: subsys/dce/dfs-admin
```

Adopting this UUID number as shown above will remedy this problem.

It is recommended that the administrator uses group types on ACLs and then adds the users to the group. In this manner, if a user is deleted you will not have to go back and adopt all of the orphaned UUIDs in ACLs. This is because the group will still be valid.

8.3.6 When Do You Need to Synchronize the FLDB and Fileset Headers?

If you receive an:

- Error message from the `fts` command aborting
- Message that you cannot find a fileset
- Initial failure FLDB entry locked

this means that your fileset needs to be resynchronized with the FLDB.

Someone may have stopped an operation with an abort signal or a File Server machine or Server process went down after someone issued an `fts` command but before the requested operation was completed. The following commands:

`fts syncfldb -s <fileserver>` examines the fileset header. It synchronizes the filesets on the specified file server to their FLDB entry on the FLDB.

`fts syncserver -s <fldb_server>` examines every FLDB entry on an FLDB Server machine and performs synchronization to the appropriate DFS file server machines.

```
root@sys1:/.:/fs>fts syncfldb -s sys3
-- done processing entry 1 of total 2 --
-- done processing entry 2 of total 2 --
FLDB synchronized with server sys3
root@sys1:/.:/fs>fts syncserv -s sys3
Server sys3 synchronized with FLDB
```

You typically use the `syncfldb` command on all the file servers that you want to synchronize. Then you use the `fts syncserver` command (one time only) on the FLDB.

Synchronizing Non-LFS Filesets

The fts syncfldb and fts syncserv commands can be used to ensure consistency of non-LFS filesets. However, because non-LFS filesets do not have fileset headers fts commands can not recreate the fileset entry using the same fileset name. A dummy name is generated and that dummy name is used to register the non_LFS fileset. You can use the fts rename command to rename the fileset back to its original name.

8.4 No Authorization Due to Expired Tickets

Tickets in DCE have a lifetime which naturally expires. If certain functions seem not to work in the DFS environment, one of the first things you must check is if the tickets are still valid. A quick way of doing this is by issuing the klist command, which lists the cached DCE tickets. The output of the klist with valid tickets looks like:

```
# klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../dino/cell_admin
        Cell:      007979f8-7d23-1c38-9eb1-10005aa83bfa /.../dino
        Principal: 00000064-7d23-2c38-9e00-10005aa83bfa cell_admin
        Group:     0000000c-7d23-2c38-9e01-10005aa83bfa none
        Local Groups:
                0000000c-7d23-2c38-9e01-10005aa83bfa none
                00000064-7d35-2c38-8a01-10005aa83bfa acct-admin
                00000065-7d36-2c38-8a01-10005aa83bfa subsys/dce/sec-admin
                00000066-7d36-2c38-8a01-10005aa83bfa subsys/dce/cds-admin
                00000068-7d37-2c38-8a01-10005aa83bfa subsys/dce/dts-admin
                00000067-7d36-2c38-8a01-10005aa83bfa subsys/dce/dfs-admin
                00000069-7d37-2c38-8a01-10005aa83bfa subsys/dce/dskl-admin

Identity Info Expires: 93/07/28:20:36:17
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_4143391a
Default principal: cell_admin@dino
Server: krbtgt/dino@dino
        valid 93/07/28:10:36:17 to 93/07/28:20:36:17
Server: dce-rgy@dino
        valid 93/07/28:10:36:17 to 93/07/28:20:36:17
Server: dce-ptgt@dino
        valid 93/07/28:15:55:23 to 93/07/28:17:55:23
Client: dce-ptgt@dino   Server: krbtgt/dino@dino
        valid 93/07/28:15:55:23 to 93/07/28:17:55:23
Client: dce-ptgt@dino   Server: hosts/sys1/dfs-server@dino
        valid 93/07/28:15:55:24 to 93/07/28:17:55:23
Client: dce-ptgt@dino   Server: dce-rgy@dino
        valid 93/07/28:16:34:21 to 93/07/28:17:55:23
```

If the ticket has expired, DFS will give you the following message:

```
dfs: ticket has expired, running unauthenticated.
```

If you then did a klist, you would see the following (in our case for the cell_admin):

```
# klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../dino/cell_admin
        Cell:      0009dbac-e70c-1d57-adc7-10005aa881fd /.../dino
        Principal: 00000064-e70c-2d57-ad00-10005aa881fd cell_admin
        Group:     0000000c-e70c-2d57-ad01-10005aa881fd <group name unknown>
        Local Groups:
                0000000c-e70c-2d57-ad01-10005aa881fd <group name unknown>
                00000064-e722-2d57-af01-10005aa881fd <group name unknown>
                00000065-e723-2d57-af01-10005aa881fd <group name unknown>
                00000066-e723-2d57-af01-10005aa881fd <group name unknown>
                00000068-e724-2d57-af01-10005aa881fd <group name unknown>
                00000067-e723-2d57-af01-10005aa881fd <group name unknown>
                00000069-e724-2d57-af01-10005aa881fd <group name unknown>

Identity Info Expires: 94/04/29:00:33:38
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_41b46673
Default principal: cell_admin@dino
```

You must then login to DCE or if you are logged in already renew your ticket. Renewing the ticket can be achieved with:

```
# kinit
```

8.5 Time out of Sync in DCE Cell

All machines participating in a DCE cell need to have their clocks set within a certain time range. This can be ensured by running the DTS in your cell. If, for example the FLDB machine′s clock is off by more than a predefined range from the other DFS machine clocks, then the DFS commands, DFS filespace access, and also many other DCE functions do not work.

How do you do a quick test if the machine clocks are correct? You can login or telnet to each of the systems and execute the date command, but this is not very elegant. A better and simple procedure is by running a small shell program, which queries the time from each machine. Execute the following shell script by either putting it into a file or by entering it directly on the command line.

```
while true
 do
   for H in sys1 sys2 sys3
    do
     echo $H
     rsh $H date
    done
    echo '-------------------------'
 done
```

The output of the previous shell will look similar to the following:

```
sys1
Wed Jul 28 16:08:21 CDT 1993
sys2
Wed Jul 28 16:08:21 CDT 1993
sys3
Wed Jul 28 16:08:22 CDT 1993
---------------------------------
sys1
Wed Jul 28 16:08:25 CDT 1993
sys2
Wed Jul 28 16:08:26 CDT 1993
sys3
Wed Jul 28 16:08:27 CDT 1993
---------------------------------
-C#
```

The use of the rsh shell requires that you maintain a /.rhosts file on all participating systems. An example of the /.rhosts follows.

```
# cat /.rhosts
sys1
sys2
sys3
```

If one of the system clocks deviates more than 10 minutes from the others, you will have to correct the time on this system. You should avoid to set time backwards, because you could create invalid entries (like future time stamps) in the CDS or FLDB. To set the time on one of the systems use the setclock command. For example, to set the time on system sys3 to the same value as on sys1 enter the following command on sys3.

```
# setclock sys1
```

Note to use setclock

The setclock command requires to have a timeserver set up or at least to have the inetd daemon running on the master system and a time program configured in the /etc/inetd.conf configuration file.

8.6 Administration List Problems

Due to reconfiguration in our cell we eventually had entries in some of the administrative lists of the form "user: 0000006d-8484-2c38-8a00-10005aa83bfa". These are the UUIDs of principals such as hosts/sys2/dfs-server as they were registered in the CDS. The bos rmadmin command can not be used to delete these users. We fixed this problem by simply removing the complete list and building a new list on the DFS system control machine. Here is an example on how to rebuild the admin.up list:

```
# mv admin.up admin.up.old
# bos addadmin -s /.:/hosts/sys1 -adminlist admin.up  \
-principal hosts/sys1/dfs-server -createlist
# bos addadmin -s /.:/hosts/sys1 -adminlist admin.up  \
-principal hosts/sys2/dfs-server
# bos addadmin -s /.:/hosts/sys1 -adminlist admin.up  \
-principal hosts/sys3/dfs-server
# bos addadmin -s /.:/hosts/sys1 -adminlist admin.up  \
-group subsys/dce/dfs-admin
```

If the upclient process is not running on a DFS server, you may have to start it. First check, if the upclient process is defined in the BosConfig file.

```
# pg BosConfig
restarttime 11 0 4 0 0
checkbintime 3 0 5 0 0
bnode simple ftserver 1
parm /opt/dcelocal/bin/ftserver
end
bnode cron backup 1
parm /opt/dcelocal/bin/fts clone -fileset sys2ft.2 -aggregate lfs.sys2.2 -server /.:/
hosts/sys2 -localauth
parm 16:10
end
bnode simple upclient.scm 1
parm /opt/dcelocal/bin/upclient -s /.:/hosts/sys1 -path /opt/dcelocal/var/dfs/admin.b
os /opt/dcelocal/var/dfs/admin.fl /opt/dcelocal/var/dfs/admin.ft
end
```

If the upclient.scm process is defined and the bosserver process is running then it is the task of the bosserver to start the upclient. You may want to check the error log file (/var/dce/dfs/adm/BosLog). Also it may be helpful to check the status of the process and eventually restart the process with the following commands:

```
# bos status -s /.:/hosts/sys2 -p upclient.scm
# bos start -s /.:/hosts/sys2 -p upclient.scm
#
```

If the upclient.scm process is not defined in BosConfig then create and start the process with:

```
# bos create -s /.:/hosts/sys2 -p upclient.scm -type simple -cmd \
'/opt/dcelocal/bin/upclient -s /.:/hosts/sys1 -path /opt/dcelocal/var/dfs/admin.bos  \
/opt/dcelocal/var/dfs/admin.fl /opt/dcelocal/var/dfs/admin.ft'
```

In the previous command, list all administration lists, which you want to have distributed from the SCM Machine. Also notice the use of the -s parameter; the first -s option indicates the host where the upclient process is to be executed; the second -s option is a parameter of the upclient command and points to the SCM machine.

Warning

A principal or group must be deleted from the DFS administrative lists before it is removed from the the DCE registry. If you fail to do that, you will run into the previously described problem.

8.7 Fileset Recovery After DCE DFS Reconfiguration

In the case of reconfiguring the DCE DFS, you can recover the aggregates and filesets that you had previously. You need to have preserved the integrity of the AIX logical volumes where the aggregates resided as well as the /usr/dce/dfs/dfstab file.

You can recover the both types of filesets: DCE LFS and non-LFS, with the following steps:

1. Make sure that the network is up and running, that means TCP/IP is up and running;

2. Make sure that DCE is configured and running (in all machines: servers and clients), that means that you have the Security Server, the CDS server or servers, the Time Servers configured and running in all machines.
3. Login into DCE as *cell_admin*.
4. Reconfigure DCE DFS.
   a. Configure the System Control Machine, if necessary
   b. Configure the File Location Database Machine
   c. Configure the File Server Machine
5. Recover the aggregates and filesets
   - When you reconfigured DFS File Servers, you should have specified to load the DCE LFS kernel extension. If you did not specify that, load the kernel extensions manually using:

     `/usr/sbin/cfgdfs -a /usr/lib/drivers/dcelfs.ext`

     And, edit the file /etc/rc.dfs to uncomment the line that contains that command.
   - Verify the /var/dce/dfs/dfstab to see if the entries are correct.
   - Export the DCE DFS aggregates, using

     `dfsexport -all`

     If you receive error messages in this step, there are two cases to consider:

     a. The DCE LFS kernel extension is not loaded

        The message will say that you cannot access the device or similar

        Load the kernel extension

     b. The data inside the aggregate is damaged

        Run the salvage program on the aggregate

        `salvage -recoveronly -aggregate <aggregatename>`
   - Synchronize the data in the aggregates and the FLDB, using

     `fts syncfldb -server <servername>`
6. Finally, configure DCE DFS clients.

After these steps you will have the DCE DFS file hierarchy available and in the same mount points you had in the previous configurations.

The last thing to be concerned about is if you also have non-LFS filesets, they will not be available yet. The `fts syncfldb` does not have enough information to restore the FLDB entries for non-LFS filesets with the original name. You should rename the fileset in the FLDB using the `fts rename` command.

Save /var/dce/dfs/dfstab

The /var/dce/dfs/dfstab that contains information of exported aggregates must be preserved to the recovery and reuse of DCE DFS aggregates.

ACL Activity

If the DCE cell was reconfigured, the existing ACLs will need changes. They will contain *incorrect* UUIDs and these entries will need to be cleaned up. The default cell and users and groups that are stale from the previous cell's configuration need to be changed or eliminated. Use the `rgy_edit adopt` command. To avoid this situation, use the same UUID when you reconfigure the cell.

Appendix A. DCE Intercell Communication Setup and ACLs

This section on intercell communication setup is provided to explain the procedures used to configure our lab setup. Our lab setup consists of two DCE cells with a DFS file tree that is visible to both.

A.1 Laboratory Configuration

Figure 56 illustrates the configuration of our machines and the software that is running on each. Note that we have a DNS nameserver in both cells although this is not really necessary.

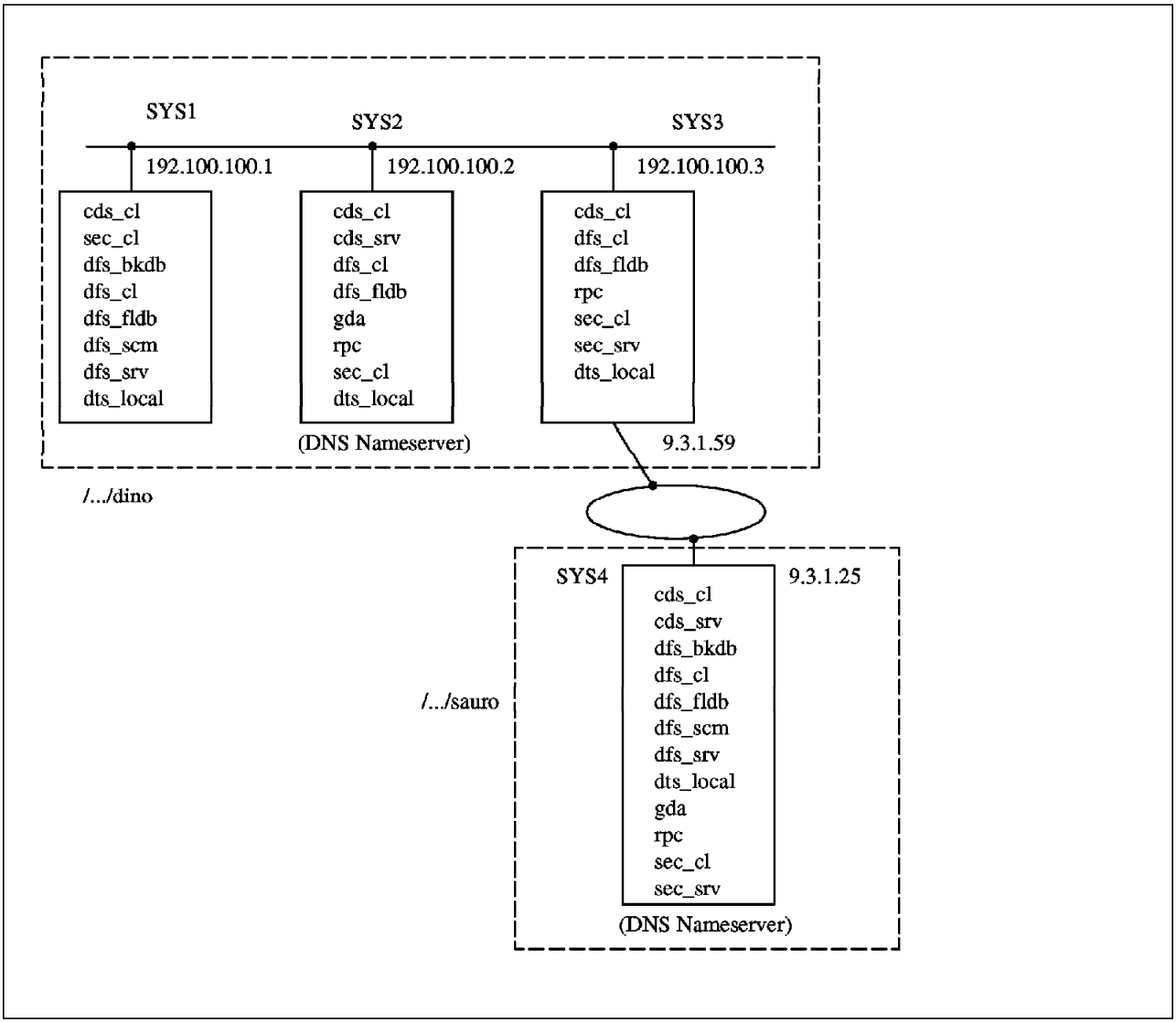


Figure 56. Our DCE Intercell Configuration

A.2 DCE GDA Configuration

To make DCE Intercell Communication possible you must have a Global Name Server (GNS). This GNS can be a DNS Server or a GDS (X.500) Server. The DCE Cell communicates with the GNS through a Global Directory Agent (GDA).

Once the Global Name Server is running, you must register the name of the cell so that the translation of the cell name to the CDS server name is possible. Once a DFS client can resolve a foreign cell name into the address of the CDS server in the foreign cell, the client can then reach the other servers in that cell and access the services that they provide.

You must have at least one GDA per DCE cell. The GDA is the interface between the CDS and the GNS.

Note that the cell names that were used in this example were not using the recommended format for registering with a DNS server. If you plan to register your cell with a DNS server, chose an appropriate name. For example, this could be: dino.dce-dfs.austin.ibm.com. Likewise, if you plan to register with a GDS server, use a GDS-style name.

A.3 Configuring Intercell Communication with a DNS Nameserver

As shown in Figure 56 on page 215, we use a DNS nameserver on sys2 to resolve the references to foreign cells. We have three systems (sys1, sys2 and sys3) in the dino cell and we have one system (sys4) in the sauro cell. We will show how to set up intercell communication between these two cells.

The following steps explain how to set up intercell communication using a DNS type of Name Server.

A.3.1 Domain Name System (DNS) Setup

We used the sys2 machine as both the Global Directory Agent (GDA) and the internet Domain Name System (DNS) name server. Since the DNS server must use full TXT records, it must be a primary nameserver. Secondary AIX.3.x nameservers do not use TXT records.

The registering is done on one machine in each cell. It does not have to be the CDS server machine, although that is most common.

The files you must modify on the DNS name server are:

- /etc/named.boot
- /etc/named.data
- /etc/named.rev
- /etc/resolv.conf

Modify the /etc/named.boot file to look like this:

```
domain  <domainname>
primary <domainname> /etc/named.data
primary in-addr.arpa /etc/named.rev
```

For example, in our setup we had:

```
domain  dce-dfs.austin.ibm.com
primary dce-dfs.austin.ibm.com /etc/named.data
primary in-addr.arpa           /etc/named.rev
```

Use the tcpip awk scripts to create /etc/named.data and /etc/named.rev. First, verify that you have the hostname of your local system and the hostname of the foreign CDS system in the /etc/hosts file. Also verify that communication between both systems are working by using the tcp/ip ping <hostname> command. Then, you can run the awk scripts that are shown below:

/usr/lpp/tcpip/samples/hosts.awk /etc/hosts > /etc/named.data
/usr/lpp/tcpip/samples/addrs.awk /etc/hosts > /etc/named.rev

The /etc/resolv.conf file must have the domain name and the nameserver.

```
domain     <domainname>
nameserver <IP address>
```

The domain name is the same as in the /etc/named.boot file. The nameserver address is the address of the nameserver in that domain.

To restart the *named* daemon:

stopsrc -s named
startsrc -s named

A.3.2 Register the Local Cell with the DNS Nameserver

Typically, you have one of your machines acting as a Primary DNS Nameserver. In our case this is sys2. The registration of the cell that contains the Primary Nameserver is done using:

mkdceregister -f /etc/named.data

This mkdceregister command modifies the /etc/named.data file and should now include information about the local cell name. Verify that the information was correctly added to the /etc/named.data file. Since we are working in the dino cell, our /etc/named.data file looks like this:

pg /etc/named.data

```
;BEGIN DCE CELL /.../dino INFORMATION
;Initial CDS server
dino.         IN      MX      1       sys2.dce-dfs.austin.ibm.com.
dino.         IN      A       192.100.100.2
dino.         IN      TXT     "1 008d2052-7ddd-1c38-85f1-02608c2fe178 Master
/.../dino/sys2_ch 008dfe82-7ddb-1c38-85f1-02608c2fe178 sys2.dce-dfs.austin.ibm.com
;END DCE CELL /.../dino INFORMATION
```

The information above means the following:

- The MX record contains the hostname of the system where the CDS server resides.
- The A record is the IP address.
- The TXT record contains:

  - The universal unique identifier (UUID) of the cell namespace in hexadecimal notation

  - The type of the replica (if master or read-only)

  - The global CDS name of the clearinghouse where the replica resides

- The UUID of the clearinghouse in hexadecimal notation
- The DNS name of the host where the clearinghouse resides

If you reconfigure your cell and your cellname or address of the CDS server changes, you must update the information in /etc/named.data with the new information.

If you received any errors, you should verify the CDS server location and the configuration information that you defined.

Before continuing, use the host command to verify that the local cell name is registered correctly.

 # host <cellname.domainname>

The response from the system should be:

```
<cellname.domainname> is ###.###.###.###
```

You can also ping the cell name. Note that the address that is returned is that of the CDS in that cell.

#ping <cellname.domainname>

You should see a response similar to:

```
PING dino.dce-dfs.austin.ibm.com: (9.3.100.3): 56 data bytes
64 bytes from 9.3.100.3: icmp_seq=0 ttl=253 time=4 ms
64 bytes from 9.3.100.3: icmp_seq=1 ttl=253 time=5 ms
                      .
                      .
                      .
```

A.3.3 Register the Foreign Cell with the DNS Nameserver

In the foreign cell that you want to comunicate with the foreign cell administrator must create a file containing the information of that cell as follows:

#cdscp show cell /.:/ as dns > /tmp/<cellname>.info
#cdscp show clear /.:/* >> /tmp/<cellname>.info

Send the file *<cellname>.info* that you have created to the foreign cell's domain nameserver machine by using the TCP/IP ftp command. (It could be the same DNS server as yours but it does not have to be.)

With the file now on the nameserver machine, register the foreign cell with the DNS nameserver as follows:

#mkdceregister -f /etc/named.data -i /tmp/<cellname>.info

This mkdceregister command takes the information about the foreign cell (contained in the /tmp/<cellname> file) and adds it to the nameserver database file (/etc/named.data). Every foreign cell that you need to communicate with has to be registered with a global nameserver. Each foreign cell will add the same type of information to it's domain nameserver.

A.3.4 Start the GDA Daemon

Start the GDA only on the CDS server machine in the local cell. On our setup, the sys2 machine is the CDS server. The following command is issued on sys2 to start the GDA:

```
# mkdce gda
```

The GDA daemon (gdad) should now be running.

When you start the GDA, it will create some RPC entries. You can verify this by using the `rpccp show mapping` command. It will show the object id, interface id, string binding and annotation.

The *gdad* daemon reads the /etc/resolv.conf file once it has started. If you make changes to the /etc/resolv.conf file, gdad must be restarted in order to reflect the changes.

If you don't have this daemon running, you are not able to change the site with the `rgy_edit site` command. You will see the following:

```
rgy_edit=> si /.../sauro

?(rgy_edit) Unable to open the registry at site "/.../sauro" - Registry
server unavailable (Registry Edit Kernel) (dce / sad)
Current site is: registry server at <none>
rgy_edit=>
```

A.3.5 Test the Unauthenticated Access

It is now possible to test the *unauthenticated* access to the foreign cell. This step allows you to verify that intercell is working from a CDS perspective. Try this command on a machine in the local cell to test access to the foreign cell. You can also run this on a machine in the foreign cell to test access to the local cell.

```
su - <guest>
cdscp show dir /.../<foreign_cell_name>
```

Since root is defined in the DCE security registry by default, root is not an unauthenticated user. Use *guest* as an unauthenticated id. The *"-"* will guarantee you are not inheriting any of the parent's environment.

If the following command shows that you have no network credentials, retrace your steps and verify that everything has been done correctly. If you try to continue and this does not work, you will be unable to create cross-cell authentication accounts.

```
$ cdscp show dir /.../dino
Warning: you have no network credentials. All requests will be unauthenticated.

                     SHOW
                DIRECTORY   /.../dino
                        AT   1993-09-02-16:44:42
Error on entity: /.../dino
Requested entry does not exist (dce / cds)
Function: dnsEnumAttr
dnsEnumAttr: partial results = /...
```

If everything has worked correctly, you should be able to show the information relating to the /...dino directory with the above command.

A.3.6 Create the Cross-Cell Authentication Accounts

When you create a cross-cell authentication account you must supply the following information:

- **Cell Name**: The name of the foreign cell with which you will establish a peer-to-peer relationship.
- **Foreign Cell Account ID and Password**: The system administrator in the foreign cell must provide you with the name and the password of an account in the foreign cell. The foreign account must have permissions required to create principals and accounts.
- **Group Name**: This is the group name associated with the account. This group has no meaning in the local cell. You must add it because it is a requirement of the registry that all accounts be associated with a group.
- **Organization Name**: This is the organization name associated with the account. It has no meaning for the account and it is not associated with any users in the foreign or local cell.
- **Account Expiration Date**: This is the time and date that both the local and foreign cell's accounts expire and the peer-to-peer relationship ends. After this date, any further authenticated communications between principals in the two cells are not allowed. The default is none.

You now are able to resolve the names of the foreign cell into its CDS server. In each cell, you need a cross-cell authentication account to allow local principals to access objects in the foreign cell as authenticated users.

When you set up the account that connects the two cells together, you do this on one machine in one of the cells. By running the rgy_edit cell subcommand, principals and accounts are automatically created in both local and foreign cells.

In our environment, from the local cell dino, you must register the foreign cell sauro as follows:

```
# rgy_edit Current site is: registry server at /.../dino/subsys/dce/sec/master
rgy_edit=> cell /.../sauro
Enter group name of the local account for the foreign cell: none
Enter group name of the foreign account for the local cell: none
Enter org name of the local account for the foreign cell: none
Enter org name of the foreign account for the local cell: none
Enter your password:  <type the password>
Enter account id to log into foreign cell with: <cell_admin>
Enter password for foreign account: <type the password>
Enter expiration date [yy/mm/dd or 'none']: (none)
Principals and Accounts have been created
rgy_edit=>
```

Warning

If you type rgy_edit and the system responds:

```
?(rgy_edit) Warning - binding is not authenticated - Cant stablish
authentication to registry (Registry Edit Kernel) (dce / sad)
rgy_edit=>
```

This means that you forgot to dce_login. You are only able to issue commands that do not require authentication.

You can create cross-cell authentication accounts to all the foreign cells that you want to access.

Do this by running the rgy_edit cell subcommand for each cell. If you want to look at the foreign cell's registry, type:

```
#rgy_edit
rgy_edit> site /.../<foreigncellname.domain>
Site changed to: registry server at /.../sauro/subsys/dce/sec/master
rgy_edit> do acc
rgy_edit> v
```

You will see the foreign cell registry and your own cell name registered.

```
        .
krbtgt/sauro [none none ]:*:106:12::::
        .
        .
krbtgt/dino [none none ]:*:101:12::::
```

You can also login as the foreign cell's cell_admin.

```
#dce_login /.../sauro/cell_admin <type password>
```

You have the privileges as the foreign cell_admin. Try `klist` to see the output.

A.3.7 Modifying Cross-Cell Authentication Accounts

By using the `rgy_edit` subcommand, you can change the account created by the cell subcommand. All authentication policy and flags that control accounts also apply to accounts for cross-cell authentication. However there are some changes that should not be performed using the standard `rgy_edit` command.

Note

Never set the account's Password-Valid flag to invalid. This will invalidate the cross-cell account.

In standard accounts, the password valid flag prompts the user to change their password at the next login. For cross-cell authentication accounts, passwords are shared by the Authentication Services in two cells. If you change one, this synchronization is destroyed and cross-cell communications ends. If you want to change the passwords shared by Authentication Services, you must reissue the `cell` subcommand to recreate the accounts and create the properly synchronized passwords.

If you delete one or both of the accounts or the account's principals, you will break the peer-to-peer relationship with the cell. If this happens, you must run the `rgy_edit cell` subcommand to recreate both accounts and restore the peer-to-peer relationship.

A.3.8 Reconfiguration of Intercell Communications

Careful cleanup must be done when reconfiguring intercell communications.

If you reconfigure one cell or you have accidentally removed a cross-cell account, you must do some cleanup before intercell communication is

established. You will detect this when you try to access the foreign site by the following command:

rgy_edit=> si /.../dino

the system will respond with

```
?(rgy_edit) Unable to open the registry at site "/.../dino" - Registry server
unavailable (Registry Edit Kernel) (dce / sad)
Current site is: registry server at <none>
```

You need to do the following:

1. In both cells, remove the surrogate authentication principal under *krbtgt/<foreign_cell>* . To do this :

   ```
   # dce_login cell_admin -dce-
   # rgy_edit
   rgy_edit>do p
   Domain changed to: principal
   rgy_edit>v          <-------------------- view current principals
           .
           .
           .
   cell_admin                                  100
   krbtgt/dino                                 101
   hosts/sys2/self                             102
   hosts/sys2/cds-server                       103
   hosts/sys2/dfs-server                       105
   hosts/sys2/gda                              108
   krbtgt/sauro                                106
   rgy_edit>del krbtgt/sauro <-------------- delete this surrogate principal
   rgy_edit>quit
   ```

2. In one cell, reissue the rgy_edit cell subcommand to generate the accounts and principals. It then recreates the peer trust between cells. See A.3.6, “Create the Cross-Cell Authentication Accounts” on page 220. Notice that both registries are modified to include the new cross-cell accounts.

3. Make sure the credentials of all principals do not have expired tickets. You can verify this by issuing the klist -e command and noting the tickets for both local and remote cells. If the principal has expired tickets he (or it) will not be able to cmomunicate with the foreign cell.

   ```
   #klist -e
   DCE Identity Information:
           Global Principal: /.../dino/hosts/sys2/self
           Cell:      00795a36-3ff8-1d0f-b041-10005aa8c755 /.../dino
           Principal: 00000066-3ff9-2d0f-b000-10005aa8c755 hosts/sys2/self
           Group:     0000000c-3ff9-2d0f-b001-10005aa8c755 none
           Local Groups:
                   0000000c-3ff9-2d0f-b001-10005aa8c755 none
                   0000006b-400f-2d0f-9101-10005aa8c755 subsys/dce/dts-servers

   Identity Info Expires: 93/10/05:19:24:38
   Account Expires:       never
   Passwd Expires:        never

   Kerberos Ticket Information:
   Ticket cache: /opt/dcelocal/var/security/creds/dcecred_ffffffff
   ```

```
Default principal: hosts/sys2/self@dino
Server: krbtgt/dino@dino
        valid 93/10/06:06:28:01 to 93/10/06:16:28:01
Server: dce-rgy@dino
        valid 93/10/06:06:28:01 to 93/10/06:16:28:01
Server: dce-ptgt@dino
   EXPIRED; was valid 93/10/05:06:38:24 to 93/10/05:08:38:24
Client: dce-ptgt@dino    Server: krbtgt/dino@dino
   EXPIRED; was valid 93/10/05:06:38:24 to 93/10/05:08:38:24
Client: dce-ptgt@dino    Server: hosts/sys2/cds-server@dino
   EXPIRED; was valid 93/10/05:06:38:24 to 93/10/05:08:38:24
Client: dce-ptgt@dino    Server: dce-rgy@dino
   EXPIRED; was valid 93/10/05:06:38:24 to 93/10/05:08:38:24
Client: dce-ptgt@dino    Server: dce-rgy@dino
   EXPIRED; was valid 93/10/05:06:59:35 to 93/10/05:08:38:24
Server: dce-ptgt@dino
        valid 93/10/06:08:39:26 to 93/10/06:10:39:26
Client: dce-ptgt@dino    Server: krbtgt/dino@dino
        valid 93/10/06:08:39:26 to 93/10/06:10:39:26
Client: dce-ptgt@dino    Server: hosts/sys2/cds-server@dino
        valid 93/10/06:08:39:26 to 93/10/06:10:39:26
Client: dce-ptgt@dino    Server: dce-rgy@dino
        valid 93/10/06:09:00:38 to 93/10/06:10:39:26
```

As you can see, there are some *EXPIRED* tickets. To remove expired tickets that, belong to an interactive user, run the `kinit` command. You may replace the credentials entirely by issuing the `kdestroy` command and then doing another `dce_login`. If the credentials belong to a non-interactive principal (like those above, which belong to the machine principal *self*), you must destroy the credentials and recreate them. For the machine principal, do this by stopping sec_clientd process and restarting it. This can be done by issuing:

```
/etc/dce.clean sec_clientd
/etc/rc.dce sec_clientd
```

Until you remove old tickets, the principal using those credentials will not be able to communicate with the foreign cell. This will be evidenced by the failure of operations such as `rgy_edit site /.../<foreign_cell>` and `cdscp show cell /.../<foreign_cell>`.

4. For more complete information about intercell communication, see the *AIX DCE Release Notes*.

A.4 DCE DFS ACLs

You can edit the appropriate DCE LFS ACLs to allow access to foreign users and groups. See 6.4, “Foreign Cell Setup” on page 188 for a discussion of ACLs for intercell access.

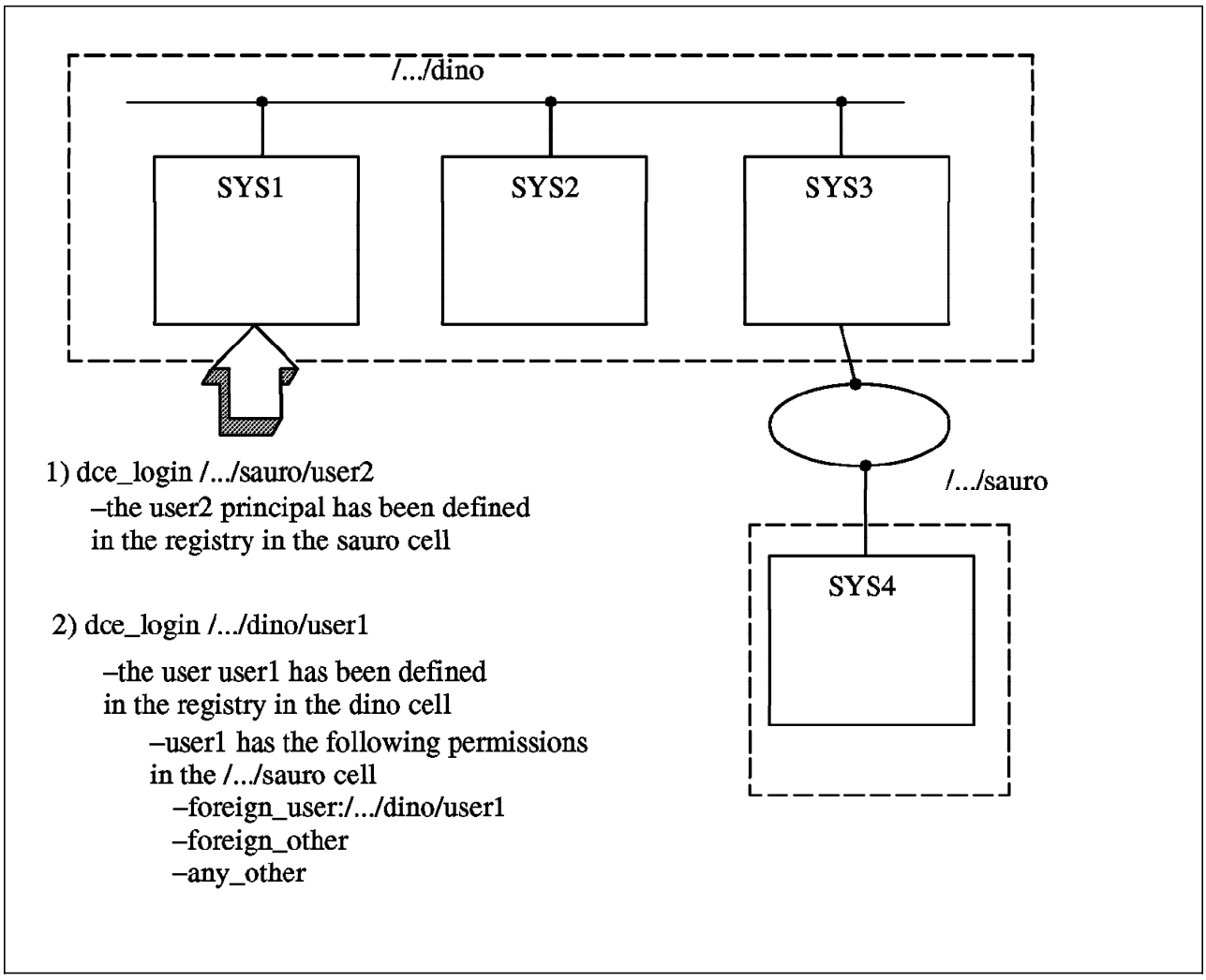


Figure 57. Our DCE Intercell Configuration

A.4.1 How to Create a File in a Foreign Cell

As shown in Figure 57, there are two ways in which users can use the filespace of a foreign cell. The first way involves a user doing a `dce_login` to his home cell from whatever cell his DCE DFS client is in. In our example, we see that user2 is doing a `dce_login` into the /.../sauro cell. The machine that he is doing the login on is defined in the /.../dino cell. For the user to be able to do a dce_login to the foreign cell, you must have established cross-cell authentication accounts as shown in A.3.6, “Create the Cross-Cell Authentication Accounts” on page 220. Once user2 has done his `dce_login /.../sauro/user2`, he is able to access his files and directories as if he were working on a DFS client defined in the /.../sauro cell. The ACL entry type that would pertain to user2 in the /.../sauro cell would be:

```
user:user2
```

In other words, he is a local user of that cell.

The second method in which a user can create a file in a foreign cell is also shown in Figure 57. In this method, user1 that has been defined in the local cell, does a `dce_login` to the local cell. In this example, user1 just does a `dce_login`

user1 from the sys1 machine. He has now authenticated to the /.../dino cell. Since cross-cell authentication accounts have been established to the /.../sauro cell, the /.../dino/user1 principal can access files and directories in the /.../sauro cell that allow the /.../dino/user1 principal access. The ACL entries types that may apply to the /.../dino/user1 principal in the /.../sauro cell are as follows:

```
foreign_user
foreign_other
any_other
```

Note that users that have not authenticated to DCE may also be able to search through directories or create files on a foreign cell if the foreign cell has the appropriate permissions set for the *any_other* ACL entry type.

A.4.2 How to Tell Who Really Owns a DFS File

Note that the owning user and group of the files will be the number associated with principals and groups that are defined in the security registry. The question in the following example is who is the owner of the test1 file?

```
# cd /:
# ls -l
-rw-r--r--   1 107      12             0 Oct 15 16:36 test1
-rw-r--r--   1 108      12             0 Oct 18 11:00 test2
-rw-r--r--   1 guest    12             0 Oct 18 11:13 test3
-rw-r--r--   1 nobody   nobody         0 Oct 18 12:19 test4
```

The owning groups and principals from the "ls -l" command are displayed as to what AIX thinks they are. AIX does not know about users and groups that have been defined in the security registry.

```
# rgy_edit
Current site is: registry server at /.../saurus/subsys/dce/sec/master
rgy_edit=> do account
Domain changed to: account
rgy_edit=> v
root [system none]:*:0:0::/::
                .
                .
krbtgt/dino [none none]:*:101:12::/::
cell_admin [none none]:*:100:12::/::
krbtgt/sauro [none none]:*:105:12::::
                .
                .
test1 [none none]:*:107:12::/home/test:ksh:
test2 [none none]:*:108:12::/home/test:ksh:
```

```
# pg /etc/passwd
        .
guest:!:100:100::/usr/guest:
nobody:!:4294967294:4294967294::/:
        .
test1:!:200:1::/u/test:/bin/ksh
test2:!:201:1::/u/test:/bin/ksh
```

For information on who the owning user is, AIX first looks at the /etc/passwd file. In our example for file1, AIX looks through /etc/passwd for the user id of 107. It does not find it so it prints out 107 as output of the `ls -l` command. If you look

up 107 in the principal domain of the security registry, you can see that it belongs to user1. The group can be determined in the same manner. This time, AIX looks in the /etc/groups file for the 12 owner. It does not find it. If you look in the group domain of the security registry, you can see that the group is *none*.

Important Recommendation

It is very important that cell administrators maintain a consistent user ID space. The passwd_import/passwd_export commands can be used to make sure that users that are defined on the local system have the same UID in the DCE Security Registry. The cell administrator should establish the policy such that when new users are created on the local system, the UID associated with their DCE principal be the same.

A.4.3 ACLs Examples

The only intercell configuration required for using DFS is setting up permissions in the cell's root file system. These permissions must be set to explicitly allow foreign users to use this filespace. Here are some examples of setting ACLs:

1. This ACL will allow all foreign users and unauthenticated users to read,execute and read the contents of directories in the remote cell's filespace root directory.

   ```
   #acl_edit /.../<cellname>/fs
   #sec_acl_edit> l
   # SEC_ACL for /.../<cellname>/fs:
   # Default cell = /.../<cellname>
   mask_obj:rwx-id
   user_obj:rwxcid
   group_obj:rwx-id
   other_obj:rwx-id
   any_other:r-x---   <-----------------
   ```

   The command to set the any_other entry to read and execute is:

   ```
   acl_edit /.../sauro/fs -m any_other:rx -l
   ```

2. This ACL will grant full access to all principals in remote cells and to unauthenticated users. They will be allowed permission to create/delete any file or directory in the DFS root directory.

   ```
   # SEC_ACL for /.../dino/fs:
   # Default cell = /.../dino
   mask_obj:rwx-id
   user_obj:rwxcid
   group_obj:rwx-id
   other_obj:rwx-id
   any_other:rwx-id
   ```

   To give these permissions to the any_other user, issue the following command:

   ```
   acl_edit /.../dino/fs -m any_other:rwxid -l
   ```

   It is the job of the cell administrator to define the right permissions for each directory and user.

3. The following is a typical default permission assigned to a directory that is owned by a specific dce_login user.

```
# acl_edit /.../sauro/fs/test
sec_acl_edit> l
# SEC_ACL for /.../sauro/fs/test:
# Default cell = /.../sauro
user_obj:rwxcid
group_obj:r-x---
other_obj:r-x---
```

If you want to add permission for a foreign group to access the file under this directory, issue:

```
# acl_edit /.../sauro/fs/test -m foreign_group:/.../dino/testgr:rwxid -l

# SEC_ACL for /.../sauro/fs/test:
# Default cell = /.../sauro
mask_obj:rwx-id
user_obj:rwxcid
group_obj:r-x---
foreign_group:/.../dino/testgr:rwx-id
other_obj:r-x---
```

If you want to add permission to allow any foreign user from a specific cell to access the file under this directory, issue:

```
# acl_edit /.../sauro/fs/test -m foreign_other:/.../dino:rx -l

# SEC_ACL for /.../sauro/fs/test:
# Default cell = /.../sauro
mask_obj:rwx-id
user_obj:rwxcid
group_obj:r-x---
foreign_group:/.../dino/testgr:rwx-id
foreign_other:/.../dino:r-x---
other_obj:r-x---
```

Appendix B. Multiple Administrative Domains

If your cell is very large there may be a need to break the cell into several smaller administrative domains. Each DFS administrative domain needs one System Control Machine (SCM). The SCM holds the master DFS administrative lists and distributes these DFS administrative lists to other DFS client and server systems. DFS client systems can be a member of several DFS administrative domains. The relationship of SCM and DFS clients is defined via the upserver/upclient processes.

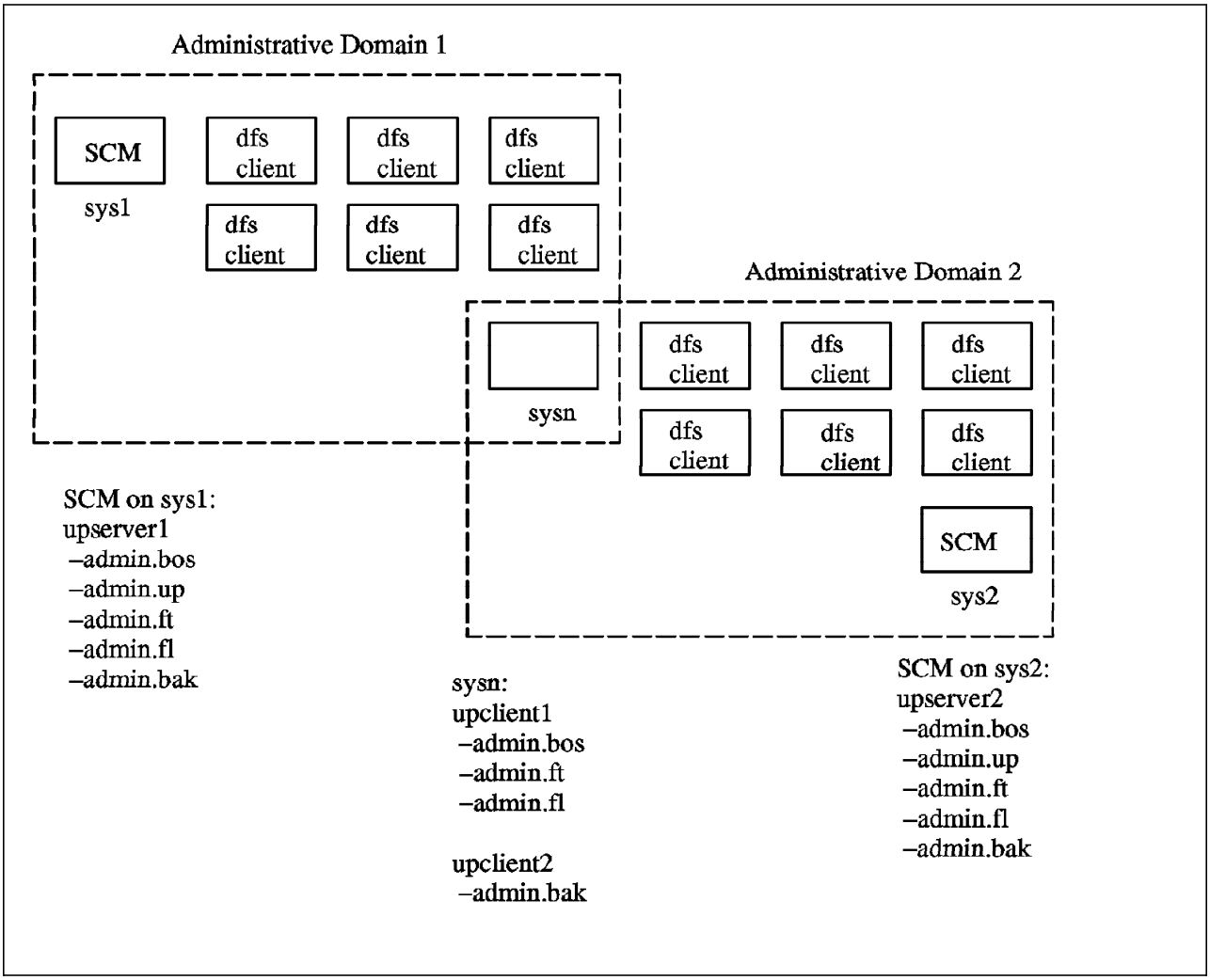


Figure 58. One DCE Cell Showing Two Administrative Domains

Figure 58 shows a cell with two DFS administrative domains. In each of these DFS administrative domains you must configure one SCM. DFS clients can belong to one or more DFS administrative domains. In case they belong to more than one SCM, you would have to start the upclient process several times with different -server parameters. The following example shows how to create upclient processes on a DFS client system (sysn) which belongs to two DFS administrative domains (sys1 and sys2).

```
# bos create -s /.:/hosts/sysn -p upclient.scm1 -type simple \
-cmd '/opt/dcelocal/bin/upclient -s /.:/hosts/sys1 \
-path /opt/dcelocal/var/dfs/admin.bos \
/opt/dcelocal/var/dfs/admin.ft /opt/dcelocal/var/dfs/admin.fl'
#
# bos create -s /.:/hosts/sysn -p upclient.scm2 -type simple \
-cmd '/opt/dcelocal/bin/upclient -s /.:/hosts/sys2 \
-path /opt/dcelocal/var/dfs/admin.bak'
```

This would request the admin.bos, admin.ft, and admin.fl lists from sys1 and the admin.bak from sys2. The DFS client *sysn* would have to be included in the admin.up list of both SCM machines.

Appendix C. Private File Server

This section will discuss the purpose of the private file server and how to configure it.

C.1 Overview of PFS

A DFS client machine can also export its local file system for use in the DFS namespace when it is configured as a DFS Private File Server machine. Unlike a regular File Server, the Private File Server's local disk is not used for data storage for an entire cell or domain. It means that the other administrative users are prohibited from creating filesets on the Private File Server. Only the owner of the machine can export data on its local disk to the global namespace.

Why do we need a DFS Private File Server? Let's suppose that you have a DFS machine which stores the data needed to be shared over the cell and the owner of that machine doesn't want that machine to be controlled by any other administrative users except himself. The purpose of a Private File Server is to exclusively allow individual users to export and control their data. Note that if the individual user is not a cell administrator he will have to work with the cell administrator to do the configuration. This is because some of the steps will require the permissions associated with being a cell administrator.

To allow only the owner of the Private File Server to control the machine, a Private File Server machine should have separate admin.bos file and admin.ft file. And these administrative files should include only the owner of the Private File Server machine. The administrative users included in such files as admin.bos and admin.ft would supersede the admin users listed in the System Control machine's admin.bos file and admin.ft file.

Figure 59 on page 232 illustrates the difference between the regular File Server machine and Private File Server machine from the viewpoint of processes running on each machine. *sys2* has an *upclient* process to retrieve administrative lists from *sys1* which is a System Control machine of the domain. *sys5* does not have a *upclient* process. As *sys5* has its own administrative lists (admin.bos and admin.ft), it does not have to retrieve administrative lists from the SCM.

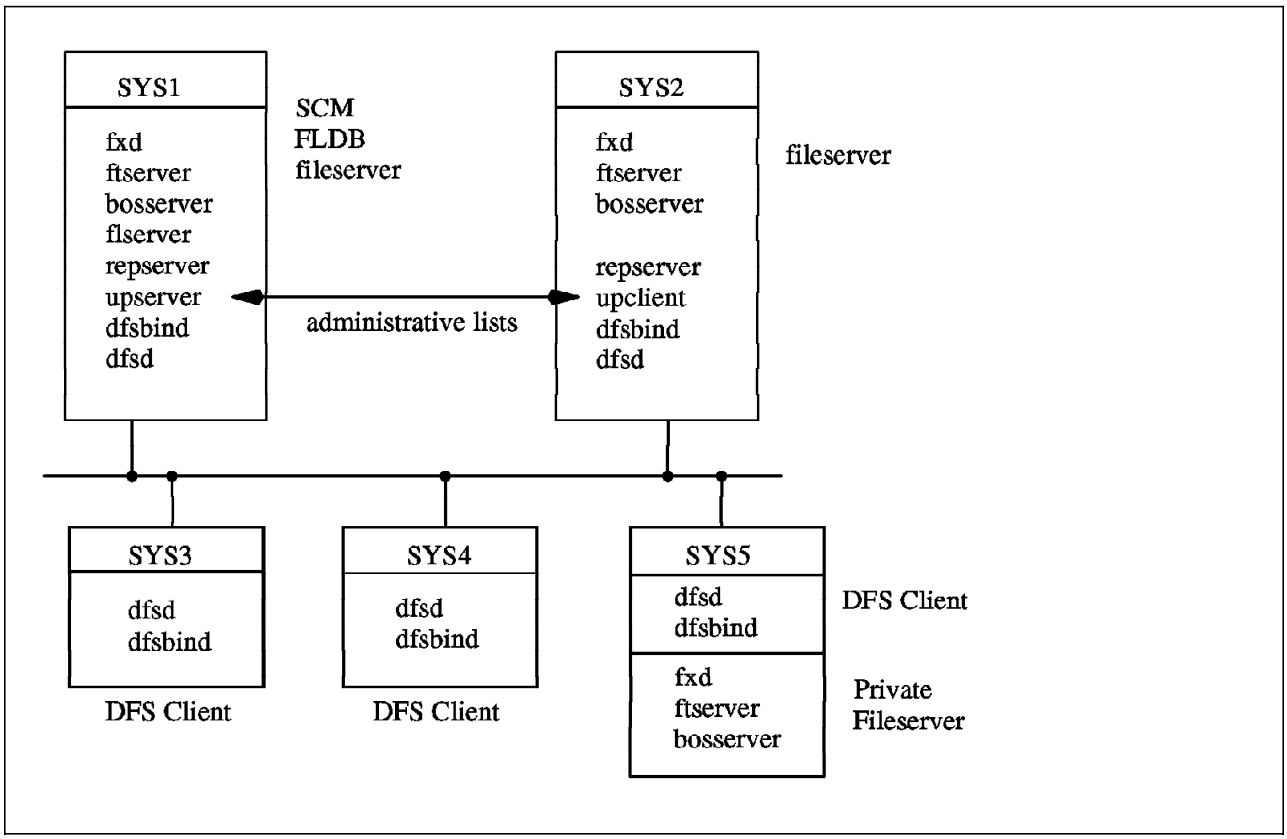


Figure 59. Processes Running on DFS File Server and Private File Server

C.2 How to Configure a DFS Private File Server

Configuring a machine as a DFS Private File Server (PFS) machine is very similar with the procedures necessary for configuring a DFS regular File Server machine. A Private File Server requires some additional tasks to create its own version of the admin.bos file and admin.ft files.

To configure a system as a PFS, perform the following steps:

Note

The machine to be configured as a DFS Private File Server must already be configured as a DFS Client machine. Therefore, the machine must run the RPC, CDS, and Security processes necessary for configuration as a DCE client machine.

1. Create a DCE user and a DCE group who are going to have authority on this machine (in this example, we will use the user *sookeun* and the group *private*.). If you require additional information on this step, refer to the *AIX DCE Administration Guide*.

2. As root, start SMIT with `mkdfssrv`

   smit mkdfssrv

   or select the following sequence of SMIT menu options:

a. *Communication Applications and Services*

b. *DCE(Distributed Computing Environment)*

c. *Configure DCE on the Local Machine*

d. *Configure DCE Servers*

e. *DFS File Server Machine*

3. For the following screen:

   - In the “Additional GROUP to administer filesets on this machine” field , specify the group’s name to which you want to assign the administration authority over the filesets on this machine. Use the group’s name which you created in the previous step (for example, *private*).
   - In the “Load LFS kernel extension?” field, type **yes**. To manipulate the DFS LFS filesets, LFS kernel extension must be loaded.
   - Leave the “DFS System CONTROL machine to get administration lists from” field blank so that this system will not retrieve the administration lists from the SCM.
   - In the “FREQUENCY to update administration lists (in seconds)” field and “LOG file for administration list updates” field, leave these fields blank.
   - If this machine has already been configured as a DCE client, the other fields are already filled in like following screen. If they are not, use the instructions in “Configuring DCE Clients” chapter of the *DCE Administration Guide* to determine the values to enter.

```
                          DFS File Server Machine

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Additional GROUP to administer filesets on this ma [private]
  chine
  Load LFS kernel extension?                          [yes]
        +
  DFS System CONTROL machine to get                   []
      administration lists from
  FREQUENCY to update administration lists (in secon []
  ds)
  LOG file for administration list updates            []
* CELL name                                           [/.../dino]
* SECURITY server                                     [sys3]
  CDS server (If in a separate network)               []
* Cell ADMINISTRATOR's account                        [cell_admin]
* LAN PROFILE                                         [/.../dino/lan-profile
]



F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Undo            F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

4. Press **Enter**

5. When prompted, enter the cell administrator’s password.

   At this time, the machine is configured as a DFS Private File Server machine.

Remember...

This procedure only configures the processes necessary for the Private File Server. Data is not exported yet from the machine to clients. For information on exporting data refer to 4.4.4, “Step 4: Exporting Data from a DFS File Server” on page 89.

6. As **cell_admin**, list the administrative lists on your system:

```
# bos lsadmin -server /.:/hosts/sys5 -adminlist admin.bos
Admin Users are: group: subsys/dce/dfs-admin
#
# bos lsadmin -server /.:/hosts/sys5 -adminlist admin.ft
Admin Users are: user: hosts/sys1/dfs-server, user: hosts/sys2/dfs-server,
user: hosts/sys5/dfs-server, group: subsys/dce/dfs-admin
```

7. Add the group you created for administrating your machine to the administrative lists on your system (in this example, *private*):

```
# bos addadmin -server /.:/hosts/sys5 -adminlist admin.bos -group
private
# bos addadmin -server /.:/hosts/sys5 -adminlist admin.ft -group
private
```

8. Remove the users and the groups from the administrative lists on your system except the group which you added in the previous step:

```
# bos rmadmin -server /.:/hosts/sys5 -adminlist admin.ft -principal
user_name
# bos rmadmin -server /.:/hosts/sys5 -adminlist admin.ft -group
group_name
# bos rmadmin -server /.:/hosts/sys5 -adminlist admin.bos -principal
user_name
# bos rmadmin -server /.:/hosts/sys5 -adminlist admin.bos -group
group_name
```

   - Replace *user_name* with the list of user, if applicable.
   - Replace *group_name* with the list of group, if applicable.

9. List the administrative lists again to make sure it includes the only group which you added in the previous step:

```
# bos lsadmin -server /.:/hosts/sys5 -adminlist admin.bos
Admin Users are: group: private

# bos lsadmin -server /.:/hosts/sys5 -adminlist admin.ft
Admin Users are: group: private
```

10. Logout from AIX, and reboot the system.

    The new administrative files will take effect only after rebooting the system.

List of Abbreviations

ACLs	Access Control Lists
AFS	Andrew File System
BOS Server	Basic OverSeer Server
BDM	Binary Distribution Machine
BK Fileset	Backup Fileset
CDS	Cell Directory Service
DCE	Distributed Computing Environment
DFS	Distributed File Service
DNS	Domain Name Service
DTS	Distributed Time Service
FLDB	Fileset database machine
FL Server	Fileset Location Server
GB	Gigabyte
GDA	Global Directory Agent
GDS	Global Directory Service
GID	Group ID
GNS	Global Name Server
IBM	International Business Machines Corporation
ID	Identification
IOC	Initial Object Creation ACL
ICC	Initial Container Creation ACL
IP	Internet Protocol
ITSC	International Technical Support Center
ITSO	International Technical Support Organization
JFS	Journaled File System
KB	Kilobyte
LFS	Local File System
MB	Megabyte
NFS	Network File System
OS	Operational System
OSF	Open Software Foundation
PFS	Private File Server
PTFs	Program Temporary Fixes
RO Fileset	Read Only Fileset
RPC	Remote Procedure Call
RW Fileset	Read Write Fileset
SCM	System Control Machine
SMIT	System Management Interface Tool
TCP	Transmission Control Protocol
TCID	Tape Coordinator ID
UDP	User Datagram Protocol
UFS	Unix File System
UID	User ID
UUID	Universal Unique Identifier
VFS	Virtual File System
VFS(+)	Extended Virtual File System
XDS	X/Open Directory Service

Index

E

F

H

I

ITSO Technical Bulletin Evaluation RED000

The Distributed File System (DFS) for AIX/6000

Publication No. GG24-4255-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____ _____

Name Address

_____ _____

Company or Organization

_____ _____

Phone No.

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

IBM International Technical Support Organization
Department 948, Building 821
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

GG24-4255-00