OSF™ DCE

OSF™ DCE User's Guide and Reference

OPEN SOFTWARE FOUNDATION

OSF™ DCE
User's Guide and Reference

Revision 1.0

Open Software Foundation

Prentice Hall, Englewood Cliffs, New Jersey 07632

Cover design: **BETH FAGAN**
Cover illustration: **STEVE LEWONTIN**

This book was formatted with troff

Published by Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-643842-3

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

Part 1C. DCE Distributed File Service

List of Figures

List of Tables

Preface

The *OSF DCE User's Guide and Reference* explains how to use the OSF^{TM} Distributed Computing Environment (DCE) services.

Audience

This guide and reference is for users who are familiar with the UNIX environment but are not familiar with DCE.

Applicability

This is Revision 1.0 of this document. It applies to the OSF^{TM} DCE Version 1.0 offering and related updates. See your software license for details.

Purpose

The purpose of this guide and reference is to provide a user with task and reference material on the DCE Directory Service, the DCE Security Service, and the DCE Distributed File Service. After reading this guide and reference, you should be able to use applications provided by DCE.

Document Usage

This document is organized into 4 parts that contain a total of 10 chapters.

- For information on the DCE Directory Service, see Part 1A.
- For information on the DCE Security Service, see Part 1B.
- For information on the DCE Distributed File Service, see Part 1C.
- For information on Security and Distributed File Service commands, see Part 2.

Related Documents

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

- *Introduction to OSF DCE*
- *OSF DCE User's Guide and Reference*
- *OSF DCE Application Development Guide*
- *OSF DCE Application Development Reference*
- *OSF DCE Administration Guide*
- *OSF DCE Administration Reference*
- *OSF DCE Porting and Testing Guide*
- *Application Environment Specification/Distributed Computing*

- *OSF DCE Technical Supplement*
- *OSF DCE Release Notes*

Typographic and Keying Conventions

This document uses the following typographic conventions:

Bold	**Bold** words or characters represent system elements that you must use literally, such as commands, flags, and pathnames.
Italic	*Italic* words or characters represent variable values that you must supply.
`Constant width`	Examples and information that the system displays appear in `constant width` typeface.
[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
...	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
<Ctrl-*x*> or ^*x*	The notation **<Ctrl-*x*>** or ^*x* followed by the name of a key indicates a control character sequence. For example, **<Ctrl-c>** means that you hold down the control key while pressing **<c>**.
<Return>	The notation **<Return>** refers to the key on your terminal or workstation that is labeled with the word Return or Enter, or with a left arrow.

Problem Reporting

If you have any problems with the software or documentation, please contact your software vendor's customer service department.

Pathnames of Directories and Files in DCE Documentation

For a list of the pathnames for directories and files referred to in this document, see the *OSF DCE Administration Guide* and the *OSF DCE Release Notes*.

Part 1

DCE User's Guide

Part 1A

DCE Directory Service

Chapter 1

Introduction to DCE Directory Service Names

The DCE Directory Service makes it possible to contact people and to use resources such as disks, print queues, and servers anywhere in the network without knowing their physical locations. The DCE Directory Service is much like a telephone directory assistance service that provides a phone number when given a person's name. The DCE Directory Service, given a unique name of a person, server, or resource, can return the network address and other information associated with that name.

The Directory Service stores addresses and other relevant information as attributes of the name. For example, attributes can contain the name of an organizational division, such as European Sales; a location, such as the first floor of Building A; or a telephone number. Users can search for a name by supplying one or more of its attributes. For example, given the search criteria of **John Smith** and **Chicago**, the Directory Service could produce a list of telephone numbers for users in Chicago named John Smith. Search capabilities are currently limited to the global part of the DCE Directory Service environment.

1.1 How People Use Directory Services

Other than DCE administrators, the people who use directory services normally do so indirectly, through an application interface. An application can interact with the DCE Directory Service on behalf of users, who create a name for a resource and subsequently refer to it by that name. The following examples, both real and hypothetical, illustrate some of the ways people can use the Directory Service:

- A user invokes a spell-checking application on a new memo. The application contains DCE Remote Procedure Call (RPC) client code on the user's local system. The RPC client contacts the DCE Directory Service for information on an available spell-checking server. The Directory Service returns the address of the server, the protocol type it uses to communicate, and a Universal Unique Identifier (UUID) that represents an interface. Using this information, the RPC client makes a remote call to the server and the server checks the spelling in the user's document. The user is unaware that use of the spell checker involved a call to the Directory Service and interaction with a remote server.
- A user logging into a system enters a name and password. The DCE Directory Service helps the login program locate an authentication server, which verifies the user's identity in an authentication database.
- A user enters a file specification. The DCE Directory Service provides the address of a DCE Distributed File Service (DFS) fileset location database, which contains the network address of a server that allows the user to access the file.
- A user enters the name of a computer conference or electronic bulletin board. The DCE Directory Service provides an address, which allows the application to connect to the conference service.
- A user enters a name or some information about a printer's capabilities. By doing this, the user can learn the printer's network address. For example, the user might want to find the address of the closest and fastest available color printer.
- A user needs information from an employee in the marketing department. The user remembers that the employee's last name is Wong, but cannot remember the first name. By entering the last name and department name in an employee locator application, the user can check the DCE Directory Service for information on all Wongs in the

marketing department and find out how to contact the employee.

- A user enters a report in a problem-tracking database. Although the database was recently moved to a new node, the user is not aware of the change because the database is always referred to by name only. The DCE Directory Service stores the current network address and provides it to the problem-tracking application and any other application that requests it.

The remainder of this chapter introduces the concept of a cell and explains how the DCE Directory Service environment works with regard to cells. The chapter describes how cells influence naming conventions and explains the difference between global and local (or cell-relative) names.

1.2 Directory Services and the Cell Environment

A cell is the basic unit of operation and administration in DCE, so understanding cells is the first step in understanding the DCE naming environment. A cell is a group of users, systems, and resources that are typically centered around a common purpose and that share common DCE services. At a minimum, the cell configuration includes one cell directory server, one security server, and three time servers.

A cell usually consists of nodes in a common geographic area (for example, on the same Local Area Network (LAN)), but geography does not necessarily determine its boundaries. It can include one system or as many as several thousand. A small organization might have only one cell, but a large multinational corporation can have many cells. Factors that can determine the configuration of cells include an organization's size, its network topology, and its needs and preferences.

Although end users may not recognize the distinction, the DCE directory components that operate within a cell and outside of a cell are different. This section introduces the following main components of the DCE naming environment and explains their relationship to the cell:

- DCE Cell Directory Service (CDS)
- DCE Global Directory Service (GDS)
- Domain Name System (DNS)

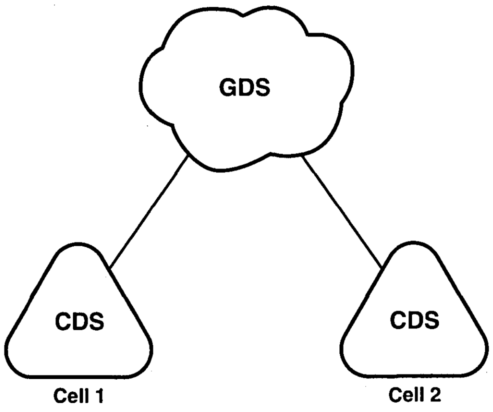
- DCE Global Directory Agent (GDA)

CDS is a high-performance distributed service that provides a consistent, location-independent method for naming and using resources inside a cell.

GDS supports the global naming environment between cells (intercell) and outside of cells (extracell). GDS is an implementation of a directory service standard known as X.500. This standard is specified by the International Organization for Standardization (ISO) 9594 and the International Telegraph and Telephone Consultative Committee (CCITT) X.500 series. Because it is based on a worldwide standard, GDS offers the opportunity for a universally interoperable global directory.

Figure 1-1 represents a hypothetical configuration of two cells that each use GDS to access names in the other cell. Names that are stored directly in GDS also are accessible from each cell. CDS is the directory service within each cell.

Figure 1–1. Cell and Global Naming Environments



DNS is a widely used existing global name service for which DCE offers support. Many networks currently use DNS primarily as a name service for Internet hostnames. Although DNS is not a part of the DCE technology offering, the DCE Directory Service contains support for cells to interoperate through DNS.

The Global Directory Agent is the DCE component that makes cell interoperation possible. GDA enables CDS to access a name in another cell by way of either of the global naming environments (GDS or DNS).

1.3 How Cells Determine Naming Environments

In addition to delineating security and administrative boundaries for users and resources, cells determine the boundaries for sets of names. Because different naming components operate in a cell and outside of a cell, naming conventions in the cell and global environments differ as well. The DCE naming environment supports two kinds of names: global names and cell-relative, or local, names. The following subsections introduce the concept of global and local names.

1.3.1 Global Names

All entries in the DCE Directory Service have a global name that is universally meaningful and usable from anywhere in the DCE naming environment. The prefix **/...** indicates that a name is global. A global name can refer to an object within a cell (named in CDS) or an object outside of a cell (named in GDS).

The following example shows the global name for an entry created in GDS. The name represents user Ellie Bloggs, who works in the administrative organization unit of the Widget organization, a British corporation. The name syntax is X.500 style; each element, except the global prefix **/...**, consists of two parts separated by an = (equal sign). The abbreviations stand for country (C), organization (O), organization unit (OU), and common name (CN).

/.../C=GB/O=Widget/OU=Admin/CN=Ellie Bloggs

For more information on X.500 naming syntax, see the *OSF DCE Administration Guide*.

The following example shows a global name for a price database server named in CDS. The server is used by the Portland sales branch of XYZ Company, an organization in the United States.

$$\overbrace{\text{/.../C=US/O=XYZ/OU=Portland}}^{\textbf{Cell Name}}\overbrace{\text{/subsys/PriceMax/price\_server1}}^{\textbf{CDS Name}}$$

As the figure illustrates, global names for entries created in CDS look slightly different from pure GDS-style names. The first portion of the name, **/.../C=US/O=XYZ/OU=Portland**, is a global cell name that exists in GDS. The remaining portion of the name, **/subsys/PriceMax/price_server1**, is a CDS name.

The cell name exists because cells must have names to be accessible in the global naming environment. The Global Directory Agent looks up the cell name in the process of helping CDS in one cell find a name in another cell.

The next example shows the global name of a host at ABC Corporation. The global name of the company's cell, **/.../abc.com**, exists in DNS.

$$\overbrace{\text{/.../abc.com}}^{\textbf{Cell Name}}\overbrace{\text{/hosts/mysystem}}^{\textbf{CDS Name}}$$

1.3.2 Local Names

In addition to their global names, all CDS entries have a cell-relative, or local, name that is meaningful and usable only from within the cell where that entry exists. The local name is a shortened form of a global name, and thus is a more convenient way to refer to resources within a user's own cell. Local names have the following characteristics:

- They do not include a global cell name.
- They begin with the **/.:** prefix.

Local names do not include a global cell name because the **/.:** prefix indicates that the name being referred to is within the local cell. When CDS encounters a **/.:** prefix on a name, it automatically replaces the prefix with the local cell's name, forming the global name. CDS can handle both global and local names, but it is more convenient to use the local name when referring to a name in the local cell. For example, the following two names are equally valid when used within the cell named **/.../C=US/O=XYZ/OU=Portland**:

/.../C=US/O=XYZ/OU=Portland/subsys/PriceMax/price_server1

/.:/subsys/PriceMax/price_server1

The naming conventions required for the interaction of local and global directory services might at first seem confusing. In an environment where references to names outside of the local cell are necessary, a few simple guidelines can help make the conventions easy to remember and use.

- Know your cell name.
- Know whether a name you are referring to is in your cell.
- When using a name that is within your cell, you can omit the cell name and include the **/.:** prefix.
- When using a name outside of your cell, enter its global syntax, including the **/...** prefix and the cell name.
- When someone asks for the name of a resource in your cell, give its global name, including the **/...** prefix.

- When storing a name in persistent storage (for example, in a shell script), use its global name, including the **/...** prefix. Local names are intended only for interactive use and should not be stored.

1.3.3 Local Filenames

When referring to pathnames of files in the local cell, you can shorten a local name even further by using the **/:** prefix. This prefix translates to the root of the cell file system. The default name of the file system root is **/.:/fs**, one level down from the root of the cell namespace. So, for example, the following are all valid ways to refer to the same file from within the **/.../widget.com** cell:

/.../widget.com/fs/smith/myfile

/.:/fs/smith/myfile

/:/smith/myfile

See Part 1C of this guide for more information on local file system abbreviations.

Chapter 2

Browsing the Cell Namespace

The Browser is a tool for viewing the content and structure of a namespace. It runs on workstations with windowing software based on the OSF/Motif™ Graphical User Interface (GUI). The Browser can display an overall directory structure as well as show the contents of directories, enabling you to monitor growth in the size and number of directories in your namespace. You also can customize the Browser so that it displays only a specific class of object names. You can see only the entries in the namespace to which you have read permission. Entries to which you do not have read permission are not displayed.

To start the Browser, enter the following command at your system prompt:

```
% cdsbrowser
```

To end a Browser session, choose **Quit** from the File pulldown menu.

When you start the Browser, an icon representing the cell root directory is the first item to be displayed in the window. Directories, soft links, and object entries all have distinct icons associated with them.

Table 2-1 shows the Browser icons and what they represent.

Table 2–1. Browser Icons

Icon	Entry Type	Icon	Entry Type
◈	Directory	◆	Object entry
◇	Clearinghouse object entry	◆	Soft link

2.1 Expanding and Collapsing Selected Directories

To expand (open) the root directory, double click on it. Double click on the expanded directory to collapse (close) it. When you expand a directory, you see all of the soft links and object entries it contains. Object entries can represent clearinghouses and any resource for which a client application creates entries in the namespace. Note that object entries representing clearinghouses are shown with a different icon than are ordinary object entries. All entries (objects, soft links, and directories) are shown indented from their parent directories.

By double clicking on single directories, you can continue expanding a particular directory path one level at a time. Other methods are available to expand all directories at once or to expand selected groups of directories.

To expand or collapse a group of directories, select them and double click. Note that, because double clicking has a toggle effect, you can expand or collapse groups of directories only one level at a time. If you double click multiple directory levels at one time, the result may be the opposite of what you expect.

To expand or collapse selected directories level by level, click on the first directory you want to select, then continue selecting directories by shift clicking (pressing <**Shift**> and clicking) on them. When you select the last directory, press <**Shift**> and double click (instead of single clicking) on it. This selects the last directory and expands or collapses all of the directories that you selected.

2.2 Expanding and Collapsing the Entire Cell Namespace

To expand all directories on all levels at once, choose the **Expand All** option from the File menu. Likewise, choose **Collapse All** from the File menu to close an expanded namespace.

Note: Use **Expand All** with care if you have a large namespace. The larger a namespace, the longer it takes to display its entire contents.

2.3 Navigating the Namespace

Once you begin expanding the namespace, it can exceed the boundaries of your Browser window, even if you enlarge the window. You can use the horizontal and vertical scroll bars and stepping arrows to scroll through the namespace.

Dragging the slider up and down the vertical scroll bar on the right side of the display window produces an index window. The index window shows the name where the slider is currently positioned in the namespace. When the index window contains the name you want to view, release the mouse button to position that name at the top of the Browser window.

In displays that are larger than the length of the window, scrolling through directory levels can produce a reference line toward the top of the window. The line orients you by showing the full directory path from the current name to the root. It also indicates that you have scrolled past other parts of the namespace that are no longer displayed.

2.4 Filtering the Namespace Display

Using the Filters menu, you can selectively display object entries of a particular class. For example, if you are interested in seeing the entries for clearinghouse objects only, choose the class **CDS_Clearinghouse** from the Filters menu. For any directory that you expand after choosing a filter, you see only names of objects whose class matches the filter. Note that soft links are still displayed because they are not object entries and only object entries can be filtered out. To reset the filter so that you can again view all object entries, choose the * (asterisk) from the Filters menu.

Part 1B

DCE Security Service

Chapter 3

User Accounts

This chapter describes

- How to log into and out of DCE
- How user identities are authenticated
- How to modify account information
- How to display registry information

Note: The descriptions in this chapter of logging into and out of DCE and of authentication reflect the default DCE methods. If your platform vendor has modified these default methods, those changes are not reflected in this document.

3.1 Logging Into and Out of DCE

Every DCE user has an account in the registry database. The account identifies the user's name and password. (Note that, besides users, servers and machines also have accounts in the registry database. The descriptions in this chapter, however, are concerned primarily with human users.)

When you log into DCE, you log into an account. To do so, you supply your name as identified in the account and the account's password. The password you supply is used in the authentication of user accounts, as described later in this chapter.

To log into DCE, use the login utility supplied by your platform vendor. Because each platform vendor can modify the utility, the exact prompts you see and the information you enter may differ according to the platform on which you are running. (The **dce_login** command, primarily used during DCE configuration, is supplied as part of DCE. See the **dce_login** reference page in the this manual for more information.) The command to exit from DCE also differs with each platform. See the specific platform documentation for more information.

3.2 Authentication of User Identities

When you log into a DCE account, the DCE Security Service checks the password you supply against the password stored for you in the registry database. If the password you supply at login is the same as the one stored for you in the registry database, the Security Service obtains your ticket-granting ticket. The ticket-granting ticket is evidence that the Security Service considers you an authenticated user and will provide you with your privilege attributes.

Privilege attributes consist of your principal name and the groups of which you are a member. They are used when you request access to objects to determine your permissions to those objects. Privilege attributes that are provided by the DCE Security Service are certified. Certified privileges are accepted by other network services. Uncertified privilege attributes may not be accepted. This means that the kinds of access to DCE objects you are allowed may differ depending on whether or not your privilege attributes are certified.

In this document, the term ''privilege attributes'' refers to certified privilege attributes unless it is specifically stated that the privilege attributes are uncertified.

3.2.1 Ticket-Granting Tickets and Tickets to Services

Your ticket-granting ticket allows you to request and receive tickets to DCE services (such as to a DFS server to read a file). The tickets that let you access DCE services are called ''service tickets.'' They tell the service that, because the DCE Security Service guarantees your identity to be as stated in your privilege attributes, you should be granted access as an authenticated user.

Both ticket-granting tickets and service tickets have lifetimes that are determined by your system administrator and by the policies in place for your installation. If the lifetime of your ticket-granting ticket has expired, you are no longer considered an authenticated user. Your access to objects other than those on the local machine is severely curtailed, and your ability to use DCE services extremely limited. To remedy this, you must reauthenticate by running the **kinit** command or by logging out and logging in again. You may want to determine your site's standard ticket lifetime and ensure you run **kinit** before your ticket-granting ticket is scheduled to expire. Section 3.2.2 instructs you in using the **kinit** command.

If your system administrator configures your account as able to renew service tickets, these tickets are renewed automatically by the DCE Security Service with no action on your part. Note, however, that the lifetime allocated to a service ticket can never exceed the time remaining on your ticket-granting ticket.

3.2.2 Using kinit to Reauthenticate

To run **kinit**, simply enter it in the following form:

% **kinit** [*principal_name*]

The *principal_name* argument is the login name of the principal whose tickets are to be renewed. If you do not enter it, **kinit** uses the login name you used to log into DCE.

The **kinit** command prompts for the password associated with *principal_name*. If you enter it correctly, your ticket-granting ticket is renewed. If you do not enter your password correctly, **kinit** displays an error message. If you get the message, run **kinit** again and enter the correct password.

Note: The **kinit** command has other options not described here. These options allow you to specify an alternate ticket cache, and request forwardable, proxiable, and renewable tickets. See the **kinit** reference page in Chapter 9 for more information.

3.2.3 Using klist to Display Privilege Attributes and Tickets

You can display your current tickets and your privilege attributes by using the **klist** command. This command lists

- Your principal name
- The groups in which you are a member
- The date and time your tickets expire
- The tickets to services that you are granted by the DCE Security Service

To run **klist** to display your current tickets, enter it with no options. (The **klist -e** option displays current and expired tickets.) The **klist** command displays three types of information: privilege attributes, expiration information, and service ticket information.

3.2.3.1 The First Part of the klist Display — Privilege Attributes

The **klist** command displays your privilege attributes. This display first lists your fully qualified principal name, followed by the UUIDs and names of your cell, your principal name (without the cell name and DCE global identifier), and all the groups of which you are a member. A sample of this section of the **klist** display follows:

```
DCE Identity Information:
   Global Principal: /.../dresden.com/music/mozart
   Cell:         5ad96550-80c4-11ca-b26c-08001e039431 /.../dresden.com
   Principal:    00000066-80c5-11ca-b600-08001e039431 music/mozart
   Group:        00000003-80c4-11ca-b201-08001e039431 composers
```

3.2.3.2 The Second Part of the klist Display — Expiration Dates and Times

The second part of the **klist** display shows the dates and time that your ticket-granting ticket, account, and password expire:

- The first line shows the date and time your ticket-granting ticket expires. Before this happens, you should reinitialize it by running **kinit** or logging in again to DCE.
- The second line shows when your account expires. If your account expires, you will be unable to log into DCE. To remedy this, your system administrator must change the account expiration date in the registry.
- The third line shows the date your password expires. When it expires, you must enter a new password before you can log into DCE.

A sample of the second part of the **klist** display follows:

```
Identity Info Expires:  91/10/03:12:07:18
Account Expires:        91/12/31:12:00:00
Passwd Expires:         91/10/31:12:00:00
```

3.2.3.3 The Third Part of the klist Display — Tickets

The third and final part of the **klist** display shows your ticket information and the name of your ticket cache. The first three tickets labeled `Server` in the following display are the tickets used after you logged in to obtain your privilege attributes. The display for all principals has these entries.

The remaining tickets labeled `Client` show your ticket-granting ticket and your service tickets. In the listing for each ticket after the word `Client`, the display shows the name of the privilege server, a server that grants your privilege attributes after your identity has been authenticated by the DCE Security Service. The name of the server to which you have tickets is shown after the `Server` entry, and the dates and times these tickets are valid are shown on the following line. For example, in the following sample display, the last line shows that the principal has a ticket to the server named **file_server**. The lifetime of this ticket is from 1:24 and 2 seconds p.m. on 10/2/91 to 12:07 and 18 seconds p.m. on 10/3/91. (The time is shown in 24-hour format.)

```
Kerberos Ticket Information:
Ticket cache: /tmp/dcecred_17a80000
Default principal: music/mozart@dresden.com
Server: krbtgt/dresden@dresden.com
   valid 91/10/02:12:07:18 to 91/10/03:12:07:18
Server:dce/rgy@dresden.com
   valid 91/10/02:12:07:20 to 91/10/03:12:07:18
Server:dce/ptgt@dresden.com
   valid 91/10/02:12:07:49 to 91/10/03:12:07:18
Client:dce/ptgt@dresden        Server:krbtgt/dresden@dresden.com
   valid 91/10/02:12:07:50 to 91/10/03:12:07:18
Client:dce/ptgt@dresden.com    Server:dce/rgy@dresden.com
   valid 91/10/02:12:07:53 to 91/10/03:12:07:18
Client:dce/ptgt@dresden.com    Server:file_server@dresden.com
   valid 91/10/02:13:24:02 to 91/10/03:12:07:18
```

3.2.4 Using kdestroy to Destroy Your Tickets

Use the **kdestroy** command to invalidate the tickets you have acquired. When you log out, your tickets are not destroyed; they remain valid until they expire. You may want to use **kdestroy** just before you log out to ensure that no valid tickets remain. To run **kdestroy**, enter it with no options.

Note: If you have the kernel-resident ticket cache, tickets are destroyed when your last process terminates. This means that it is generally not necessary to run **kdestroy** at logout.

3.2.5 Changing User Identities

The **su** command allows you to assume a different identity without logging off and on again. When you run **su**, you must supply an account name and password. When you do this, you are given a new process that is authenticated as the new identity and obtains the identity's privilege attributes. Any tickets granted are granted to this new identity. The tickets from your previous identity are valid until they expire (unless you destroy them with the **kdestroy** command), but they are not available to the new identity.

3.3 Modifying Account Information

Information about your account is maintained by your system administrator, using the DCE Security Service **rgy_edit** command. Your use of this command is likely limited to viewing account information (described later in this chapter). However, you can use the **chpass** command to change your

- Password
- Home Directory
- GECOS Information
- Login Shell

The changes you make with **chpass** are automatically reflected in the registry database. See the **chpass** command reference page in Chapter 9 for information on the **chpass** command.

3.4 Displaying Registry Information

If your system administrator sets the appropriate permissions, you can view account information in the registry database. This section briefly describes how to invoke **rgy_edit** and use it to display accounts, principals, groups, and organizations. For detailed information on using **rgy_edit** to view, create, and modify registry objects, see the *OSF DCE Administration Guide*.

3.4.1 Invoking and Exiting from the rgy_edit Command

To invoke the **rgy_edit** command, enter the following at the system prompt:

% dceshared/**bin/rgy_edit**

The **rgy_edit** command displays the **rgy_edit** prompt:

```
rgy_edit=>
```

To exit from **rgy_edit**, enter

```
rgy_edit=> quit
```

3.4.2 Displaying Accounts

Use the **rgy_edit view** command (abbreviated to **v**) in the following manner to display account information:

1. Invoke **rgy_edit** at the system prompt:

   % *dceshared*/**bin/rgy_edit**

   The **rgy_edit** command displays the **rgy_edit** prompt:

   ```
   rgy_edit=>
   ```

2. Change to the account domain:

   rgy_edit=>**do account**

3. Enter the **view** subcommand and the name of the principal whose account you want to display. (If you do not enter a principal name, **rgy_edit** displays all accounts in the cell.) The following example displays the account for the principal **mahler**:

rgy_edit=>**v mahler**

```
mahler[symphonists classic]:6XPbd13hiftzTE:24583:12::/fugue/mahler::
rgy_edit=>
```

4. If you enter the **view** subcommand with the **-f** option, you can display the account's administrative information as follows:

rgy_edit=> **v mahler -f**

```
mahler[symphonists classic]:6XPbd13hiftzTE:24583:12::/fugue/mahler::
 created by: /.../dresden.com/root      1991/06/11.19:57
 Changed by: /.../dresden.com/root      1991/07/09.19:57
  password is: valid, was last changed: 1991/07/09.00:41
  Account expiration date: 1991/11/01.10:00
  Account MAY be a server principal
  Account MAY be a client principal
  Account is: valid
  Account CAN NOT get post-dated certificates
```

```
    Account CAN get forwardable certificates
    Certificates to this service account MAY be issued
          via TGT authentication
    Account CAN get renewable certificates
    Account CAN NOT get proxiable certificates
    Account CAN NOT have duplicate session keys
    Good since date: 1991/06/11.19:57
    Max certificate lifetime: 24
    Max renewable lifetime: 168
  rgy_edit=>
```

3.4.3 Displaying Groups

Use the **rgy_edit view** command (abbreviated to **v**) to display group information by performing the following steps:

1. At the **rgy_edit** prompt, change to the group domain:

   ```
   rgy_edit=>do group
   ```

2. Enter the **view** subcommand and the name of the group to display. (If you do not enter a group name, **rgy_edit** displays all groups.) The following example displays the group **symphonists**. The display includes the group name and UNIX number.

   ```
   rgy_edit=> v symphonists

   symphonists 193
   ```

3. To display members of a group, enter the **view** subcommand with the **-m** option. The following example displays members of the group **symphonists**:

   ```
   rgy_edit=> v symphonists -m

   symphonists 193
     3 members: brahms, britten, mahler
   ```

3.4.4 Displaying Organizations

Use the **rgy_edit view** command (abbreviated to **v**) to display organization information by performing the following steps:

1. At the **rgy_edit** prompt, change to the organization domain:

   ```
   rgy_edit=>do org
   ```

2. Enter the **view** subcommand and the name of the organization to display. (If you do not enter an organization name, **rgy_edit** displays all organizations.) The following example displays the organization **classic**. The display includes the organization name and UNIX number.

   ```
   rgy_edit=> v classic

   classic 12
   ```

3. To display members of an organization, enter the **view** subcommand with the **-m** option.

3.4.5 Displaying Principals

Use the **rgy_edit view** command (abbreviated to **v**) to display principal information by performing the following steps:

1. At the **rgy_edit** prompt, change to the principal domain:

   ```
   rgy_edit=>do principal
   ```

2. Enter the **view** subcommand and the name of the principal to display. (If you do not enter a principal name, **rgy_edit** displays all principals.) The following example displays the principal **mahler**. The display includes the principal name and UNIX number.

   ```
   rgy_edit=> v mahler

   mahler 24583
   ```

3. To display all the groups of which the principal is a member, enter the **view** subcommand with the **-m** option. The following example displays the groups of which **mahler** is a member:

   ```
   rgy_edit=> v mahler -m

   mahler 24583
    Member of 1 group:
    symphonists
   ```

Chapter 4

Authorization

You can control access to DCE objects by using an authorization mechanism called an Access Control List (ACL). ACLs are associated with files, directories, CDS entries, and registry objects. They can be implemented also by arbitrary applications to control access to their internal data objects. Each ACL consists of multiple ACL entries that define who is authorized to do what to the object, specifically

- Who can use the object
- What kinds of access those principals have to the object

This chapter gives a general overview of DCE authorization and describes how to use the **acl_edit** command to maintain ACLs. For detailed information on how a specific DCE component implements the ACL authorization mechanism, see the component's section in the *OSF DCE Administration Guide*.

Note: In the discussions of DCE authorization in this chapter and the chapters that follow, the term ''user'' is analogous to principal. A principal can be a human user, server, or a machine.

4.1 Authorization Overview

An ACL contains a list of entries that specify the principals who can access an object and the operations those principals can perform. The principals can be named specifically or be members of a group that is identified in the ACL entry. The ACL is associated with the object it protects. The operations a principal can perform are specified by permissions.

Although UNIX permissions, called ''permission bits,'' can be set only for file system objects (files and directories), in DCE many objects can have ACLs and be assigned permissions. DCE ACLs control access to objects managed by DCE components, like the Distributed File Service, the Security Service, and the Directory Service. ACLs for the Registry Service (the Security component that controls accounts) can, for example, authorize certain principals to change all of the information associated with an account, authorize other principals to change only a subset of the information associated with accounts, and restrict other principals from changing any of the information associated with accounts.

In addition to the standard DCE components, ACLs can control any object for which an ACL Manager (described later in this chapter) has been implemented. ACLs can be associated with user-written applications to protect access to the use of the application itself, the files in the application, and even fields in those files.

In UNIX systems, because only file system objects are protected by permission bits, a standard set of permissions (read, write, and execute) can be used. In DCE, however, the names and meanings of the permissions themselves can be defined individually for each type of DCE object. For example, the file system uses the familiar read, write, and execute permissions, but the Registry Service uses a more extensive set of permissions specifically tailored to its authorization requirements.

4.1.1 ACL Interpretation

Part of the information associated with an account is the principal name and the group (or groups) associated with the principal name in the account. The UUIDs that represent the principal's name and group names are known as the principal's privilege attributes. If the principal's privilege attributes have been supplied by the Authentication Service, they are said to be "certified privilege attributes." Principals without certified privilege attributes are allowed only unauthenticated access to objects. Unauthenticated access, if it is allowed at all, is generally more restrictive than authenticated access.

4.1.1.1 Privilege Attributes Inherited by Processes

Processes created or spawned by a principal inherit the principal's privilege attributes. For example, if you log in, are authenticated, and start an application, the application you start inherits your authenticated privilege attributes and runs as though it were you. The application's permissions for any given object are the same as your permissions. Processes spawned by the application carry your identity and pass it down to processes they start.

Note: Changing the **setuid** permission bit changes only the local identity under which an executable file runs, not the network identity.

Some servers are written to run as separate authenticated principals. For these servers, your system administrator creates an account in the registry database. After you start these servers, the server process performs the equivalent of a user login, receives its privilege attributes, and runs under its own identity, not yours.

4.1.1.2 Granting Access

When a principal requests access to a DCE object associated with an ACL, the object's ACL Manager compares the principal's privilege attributes with the entries in the object's ACL. It does this simply by reading through the list of ACL entries. The Manager grants the access permissions in the ACL entries that match the principal's privilege attributes. To do this, the ACL Manager uses the access check algorithm, described in Section 4.12. If the permissions allow the requested mode of access, the principal gains access; if not, access is denied.

4.1.2 ACL Managers

One ACL Manager may support multiple Manager types, each Manager type implementing a different permission set. For example, access to the registry is controlled by five ACL Manager types, each controlling ACLs for a different type of object in the registry database. The **acl_edit** command that adds and modifies ACL entries has a subcommand to list the permissions specific to the ACL associated with the named object.

All of the elements of ACLs described in this chapter are available to ACL Managers; however, each Manager may implement all or only a subset of the elements. For information on how ACLs are used by specific DCE components, consult that component's section of the *OSF DCE Administration Guide*.

4.1.3 Extended Protection

In UNIX, permission bits are set for each file system object's owner and group. All other principals are grouped into the general category of ``other.'' The permission bits set for ``other'' apply to all principals except the object's owner and the members of the object's group. ACLs, on the other hand, extend this somewhat restrictive UNIX protection mechanism and give finely tuned control over access to objects.

In addition to entries similar to UNIX owner, group, and other, DCE permissions can be set for

- Specific individual principals in the local cell and in foreign cells
- Specific individual groups in the local cell and in foreign cells
- All other principals in a specific foreign cell for whom individual permissions have not been set
- All principals in any cell who have been authenticated by the DCE Authentication Service

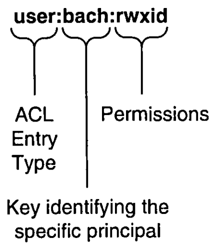
ACLs also provide a masking capability and a method for integrating protections from DCE versions that are different from the current version.

4.2 ACL Entries

ACL entries are of the form

type[*:key*]*:permissions*

The following sample ACL entry sets permissions for a principal in the local cell, named **bach**. The ACL entry type is **user**, the key is **bach,** and the permissions are **rwxid**. Each entry component is separated by a : (colon). (Note that not all types of ACL entries require you to enter a key.)



ACL entry types let you define entries for

- Principals and groups
  — Principals and groups in the local cell

  - Principals and groups in foreign cells
  - All principals in the local cell for whom individual ACL entries have not been created, and all principals in all foreign cells for whom individual ACL entries have not been created

- Masks used for authenticated and unauthenticated users
- As-yet-undefined entry types that can be copied and displayed (if not interpreted) by dissimilar DCE releases.

4.2.1 ACL Entry Types for Principals and Groups

This section describes ACL entry types for principals and groups. Note that, if any principal or group is not authenticated, the permissions in the entry are further constrained by the **unauthenticated** mask (described later in this chapter). Note also that all entries for authenticated principals, except **user_obj** and **other_obj** entries, are further constrained by the **mask_obj** mask (also described later in this chapter). The descriptions of the entries use the following syntax variables:

- *prinicipal_name* is the name used by the principal to log in.
- *group_name* is the name of the group defined in a registry database.
- *cell* is the global pathname of the foreign cell in the format */.../name*.
- *permissions* are the permissions made available by the object's ACL Manager.

The ACL entry types for principals and groups are as follows:

- The **user_obj** types establishes permissions for the object's real or effective user. The ACL entry format is

  user_obj:*permissions*

- The **group_obj** type establishes permissions for members of the object's real or effective group. The ACL entry format is

  group_obj:*permissions*

- The **other_obj** type establishes permissions for all others in ACL's default cell, unless they are specifically named in ACLs of entry type **user**, are members of a group named in an ACL with an entry type of **group**, or match the principal indicated by the **user_obj** or **group_obj** entry. The ACL entry format is

  other_obj:*permissions*

  Note: The keys for **user_obj** and **group_obj** ACLs are not named in the ACL itself, but somewhere else. For example, in the file system, a **user_obj** entry refers to the file/directory's owner.

- The **user** type establishes permissions for a specific principal in the default cell. You must identify the principal by supplying a principal name in the ACL entry. The ACL entry format is

  user:*principal_name***:***permissions*

- The **group** type establishes permissions for members of a specific group in the default cell. You must identify the group by supplying a principal name and cell name as a key. The ACL entry format is

  group:*group_name***:***permissions*

- The **foreign_user** type establishes permissions for a specific principal in a foreign cell. You must identify the principal by supplying a principal name and cell name as a key. The ACL entry format is

  foreign_user:*cell_name/principal_name***:***permissions*

- The **foreign_group** type establishes permissions for a specific group in a foreign cell. You must identify the group by supplying a group name and a cell name as a key. The ACL entry format is

  foreign_group:*cell_name/group_name***:***permissions*

- The **foreign_other** type establishes permissions for others in a specific foreign cell, unless they are specifically named in ACL entries of entry type **foreign_user** or are members of a group named in an ACL entry of type **foreign_group**. You must identify the foreign cell by supplying a cell name as a key. The ACL entry format is

  foreign_other:*cell_name***:***permissions*

- The **any_other** type establishes permissions for all others in all foreign cells, unless they are specifically named in ACL entries of entry type of **foreign_user**, are members of a group named in an ACL entry of type **foreign_group**, or are principals in a cell named in an ACL entry of type **foreign_other**. The ACL entry format is

  any_other:*permissions*

4.2.2 ACL Entry Types for Masks

Masks in ACL entries establish maximum permissions that can then be applied to individual ACL entries. There are two masks: the **mask_obj mask** and the **unauthenticated mask**. Only permissions given in an ACL entry and in its associated mask are granted. For example, if the ACL entry specifies **rwx** permissions and the mask associated with the ACL entry specifies only the **x** permission, only the **x** permission is granted.

The **mask_obj** mask (entry type **mask_obj**), if it exists, applies to all entry types except **user_obj** and **other_obj**. In addition, it does not apply to unauthenticated access. The unauthenticated mask (entry type **unauthenticated**) is applied to the permission set acquired from all entries matched by an unauthenticated principal.

4.2.3 Specifying the Default Cell Name

The default cell name applies to principals and groups identified by **user_obj**, **group_obj**, **other_obj**, **user**, and **group** ACL entry types. It identifies the cell in which the principal or group is registered. (All other entry types are for principals and groups in foreign cells and thus specify the name of the cell explicitly.)

The default cell is usually the local cell, but you can use the **acl_edit ce[ll]** subcommand to name the default cell. (Use the **l[ist]** subcommand to display the name of the default cell name.) The default you set stays in place until you change it with another **ce[ll]** subcommand. The primary use of the default cell is to allow you to copy ACLs to a cell different from the one they were created in.

4.2.4 Specifying Principals and Cells

When you create ACLs with an entry type of **user**, **group**, **foreign_user**, **foreign_group**, and **foreign_other**, you must specify the principal or group name as a key.

For **foreign_user** and **foreign_group** entries, use the principal's or group's fully qualified name in the following form:

cell_name/principal_name

The *cell_name* consists of the DCE global prefix (**/...**) followed by a slash, followed by the name of the cell. For example, for the principal named ''bach'' at the cell named ''dresden.com,'' you enter

/.../dresden.com/bach

For an entry type of **foreign_other**, you need only specify the cell name. For example:

/.../dresden.com

Note: The fully qualified principal names you enter for **acl_edit** are different from the principal names you enter for other DCE Security Service commands, such as **chpass** and **rgy_edit**. For all other Security Service commands, you can supply only the principal name and not the global prefix and cell name. Other Security Service commands can supply the required context for the principal name you enter because they work only with Security Service components. The **acl_edit** command, however, works with objects from all DCE services. These objects may possibly exist in different namespaces. You must, therefore, supply the principal's fully qualified pathname as described previously.

4.2.5 Accruing Group Permissions

Principals are added as members of groups by using the Registry Service editor (**rgy_edit**) to add the principal name to the group's membership list. The collection of groups in which a principal is a member is called the principal's ``project list.'' A principal's group access permissions to an object are the permissions of the project list, the combination of all groups in which the principal is a member and for which an entry is created in the ACL.

For example, suppose an ACL contains the following entries:

```
user_obj:crwxid-
group_obj:crwx---
other_obj:-r-----
group:composers:crwx---
user:bach:crwx---
user:mozart:crwx---
group:performers:--w-idt
```

User **cole** is a member of the group **composers** and the group **performers**. Because **cole** accrues permissions from both groups, his access permissions are **crwxidt**. (The DCE Security Service provides a method to prevent a group from being included in a project list, thus preventing the group's permissions from being accrued as part of the project list. See the *OSF DCE Administration Guide* for more information.)

4.2.6 ACL Entry Types for Dissimilar DCE Releases

The **extended** entry type provides a generic format for ACL entries that allows future DCE releases to implement new ACL entry types. Because the new types are ''packaged'' in the generic format of the **extended** entry, earlier DCE releases can copy, display, and print the new entry types even if they cannot interpret their meaning.

You can create an **extended** ACL entry by using the **acl_edit assign** command. (See Section 4.5 for information on how to use **assign**.) To do so, you first enter the original entry into a file. Once the file is created, you can use it as the file assigned to the ACL entry by the **assign** subcommand. ACL Manager types may support only a subset of the available ACL entry types listed in this document. To copy **extended** entries from the domain of one ACL Manager to the domain of another ACL Manager, both ACL Managers must support the **extended** entry type. The extended entries cannot be modified, but they can be deleted.

An **extended** ACL entry has the following form:

extended: *uuid.ndr.ndr.ndr.ndr.number_of_bytes***:***data***:***permissions*

where:

extended	The ACL entry type.
uuid	A UUID that identifies the entry type of the extended ACL entry. (This UUID can identify one of the ACL entry types described in this document or an as-yet-undefined ACL entry type.)
ndr.ndr.ndr.ndr	A Network Data Representation (NDR) format label (in hexadecimal format and separated by periods) that identifies the encoding of data.

number_of_bytes	A decimal number that specifies the total number of bytes in *data*.
data	The ACL data in hexadecimal format. (Each byte of ACL data is two hexadecimal digits.) The ACL data includes all of the ACL entry specification except the permissions. The ACL data is not interpreted; it is assumed that the ACL Manager to which the data is being passed can understand that data.
permissions	The permissions to be granted by the entry.

4.3 Using the acl_edit Command

The following subsections provide general instructions for using **acl_edit**. They describe how to invoke **acl_edit** and use the **help** facility, and list the **acl_edit** subcommands.

The **acl_edit** command creates, modifies, and displays ACL entries. Using this tool, you can

- Create and modify ACL entries for DCE objects in the local cell and foreign cells. (Note that when objects are created they are associated with an ACL and initial ACL entries. See Section 4.9 for more information.)
- Display the permissions implemented for an object by the object's ACL Manager.
- Create and modify masks used to restrict allowable permissions.

Note: Standard UNIX tools that display and manipulate UNIX modes have an effect only on the ACLs established for the file system. See Part 1C for details of the interaction between UNIX tools and file system ACLs.

4.3.1 Invoking the acl_edit Command

You can run the **acl_edit** command in interactive mode, in which the system prompts you for needed information, or in command-line mode, in which you enter all the information on the command line. In command-line mode, you can perform only one **acl_edit** operation at a time; however, you may find command-line mode useful for creating shell scripts that execute a sequence of **acl_edit** subcommands. Note that, if you have a GUI and a terminal capable of using the GUI, interactive-mode **acl_edit** runs the GUI unless you specify otherwise with the **-ngui** option.

4.3.1.1 Invoking acl_edit in Interactive Mode

To invoke **acl_edit** in interactive mode, enter the following at the system prompt:

% dceshared **/bin/acl_edit** *pathname*

where:

pathname Specifies the object whose ACL is to be displayed or modified. If the object is in another cell, you must enter the fully qualified pathname.

For example, to edit the object named **opus** in the file system named **my_filesystem**, use a pathname like

/.../dresden.com/my_filesystem/opus

The **acl_edit** command displays the following prompt:

```
sec_acl_edit>
```

4.3.1.2 Invoking acl_edit in Command-Line Mode

To invoke **acl_edit** in command-line mode, enter the following at the system prompt:

% **acl_edit** *pathname command_line_subcommand* [*acl_entry*]

where:

pathname Specifies the object whose ACLs are to be displayed or modified. If the object is in another cell, you must enter the fully qualified pathname.

For example, to edit the object named **opus** in the file system named **my_filesystem**, enter

/.../dresden.com/my_filesystem/opus

command_line_subcommand
Specifies the **acl_edit** subcommand to be executed.

acl_entry Specifies the ACL entry in the form:

type[:*key*]:*permissions*

For example, to set **rwx** permissions for user **bach** to the DCE object named **opus** in the local cell, enter the following at the system prompt:

% **acl_edit opus -m user:bach:rwx**

4.3.2 Invocation Options for acl_edit

When you invoke **acl_edit** in interactive or command-line mode, you can supply options that let you edit initial creation ACLs and specify that **acl_edit** should resolve pathnames to a CDS leaf object. Table 4-1 lists these options and where to find full explanations of these capabilities.

Table 4–1. The acl_edit Command Invocation Options

Option	Purpose
-ic	To edit a container object's Initial Container ACL. See Section 4.9.
-io	To edit a container object's Initial Object ACL. See Section 4.9.
-e	To manipulate the ACL associated with a CDS object whose pathname resolves to a leaf object in the CDS namespace and an object in the namespace of some other DCE component. See Section 4.10.
-n	To specify that a new mask should not be calculated. This option is useful only for objects that support the **mask_obj** entry type and that recalculate a new mask after they are modified.
-c	To specify that when an object's **mask_obj** type entry is created or modified, it should have permissions equal to the union of all entries other than type **user_obj**, **other_obj**, and **unauthenticated**. The creation or modification is done after all other modifications to the ACL are performed. The new mask is set even if it grants permissions previously masked out. It is recommended that you use this option only if not specifying it results in an error. This option is useful only for objects that support the **mask_obj** entry type and that recalculate a new mask after they are modified. If you specify the **-c** option for an ACL that does not support the **mask_obj** entry type, **acl_edit** returns an error when you attempt to save the ACL, aborting all subcommands supplied on the command line.

4.3.3 Exiting from acl_edit

To end the **acl_edit** session and save the changes you specified in the session, use the following **exit** command:

```
sec_acl_edit> exit
```

To end the **acl_edit** session without saving the changes you specified, use the following **abort** command:

```
sec_acl_edit> abort
```

4.3.4 Using the acl_edit Help Facility

In interactive mode, the **acl_edit help** subcommand displays help information. If you enter **h** or **?**, **acl_edit** displays a list of all subcommands and available topics. For example:

```
sec_acl_edit> h

Known commands are:
ab[ort]         as[sign_file]   co[mmit]        d[elete]
e[xit]          g[et_access]    h[elp]          k[ill_entries]
l[ist]          m[odify]        p[ermissions]   ce[ll]
sec_acl_entry   su[bstitute]    t[est_access]   ?
```

(The **sec_acl_entry** topic displays help information for ACL entries.)

If you enter **h** and a subcommand name, **acl_edit** displays information about the command. For example:

```
sec_acl_edit> h as

assign_file --
  Assign the sec_acl entries contained in the specified file
  to the object
Usage:
as[sign] FILENAME
```

4.3.5 The acl_edit Subcommands

Tables 4-2 and 4-3 describe the **acl_edit** subcommands available to you, depending on which mode (command-line or interactive) you choose to operate in.

Table 4–2. Command-Line Mode acl_edit Subcommands

Command	Function
-m *acl_entries*	Adds the new entry (or entries) specified by *acl_entry* or changes the permissions of an existing entry to those specified in the ACL entry.
-d *acl_entries*	Deletes the entry (or entries) identified by *acl_entry* from the ACL associated with the specified object.
-s *acl_entries*	Replaces (substitutes) an ACL entry with *acl_entries.* All existing entries are removed and replaced by the newly specified entry.
-f *file*	Assigns the ACL information contained in *file* to the object. The *file* argument specifies an ASCII file created using your editor of choice.
-k	Removes all entries, except the entry of type **user_obj**, if it exists.
-l	Lists the ACLs associated with the specified object.
-p	Purges all masked permissions (before any other modifications are made). This subcommand is useful only for ACLs that contain an entry of type **mask_obj**. Use it to prevent unintentionally granting permissions to an existing entry when a new mask is calculated as a result of adding or modifying an ACL entry.

Table 4–3. Interactive Mode acl_edit Subcommands

Command	Function
?	Displays the available **acl_edit** subcommands.
ab[ort]	Exits **acl_edit** without saving the changes to the object's ACL.
as[sign] *filename*	Applies the ACL entries in *filename* to the specified object. The *filename* argument specifies an ASCII file created in the editor of your choice or by the **acl_edit l** or **-l** subcommand.
ce[ll] *name*	For ACL entries whose name is scoped by the local cell (the **user_obj**, **group_obj**, **other_obj**, **user**, and **group** entries), sets the default cell name. The cell name stays in place until you run the subcommand again to change it.
co[mmit]	Saves all changes made since **acl_edit** was invoked, without exiting from **acl_edit**.
d[elete] *acl_entry*	Deletes the specified ACL entry from the specified object.
e[xit]	Exits from **acl_edit**, saving the changes to the object's ACL.
g[et_access]	Displays the permissions granted in the specified object's ACL to the principal that invoked **acl_edit**.
k[ill_entries]	Removes all ACL entries except the **user_obj** entry if it exists.
l[ist]	Lists the entries in the specified object's ACLs.
m[odify] *acl_entry*	Adds a new ACL entry or replaces an existing ACL entry. This command affects a single ACL entry. To add or replace all of an object's ACL entries, see the **substitute** subcommand. For objects that calculate a new mask when their ACLs are modified, the **-n** option specifies that a new mask should not be calculated; the **-c** option specifies that the object's **mask_obj** type entry should have permissions equal to the union of all entries other than type **user_obj**, **other_obj**, and **unauthenticated**. The mask is

Command	**Function**
	calculated after the ACL is modified. The new mask is set even if it grants permissions previously masked out. It is recommended that you use the **-c** option only if not specifying it results in an error. If the new mask unintentionally grants permissions to an existing entry, the modify operation causing the mask recalculation will abort with an error unless you specify either the **-c** or **-n** option.
su[bstitute] *acl_entry* [*acl_entry*...]	Replaces all ACL entries associated with the specified object with the one or ones specified. This subcommand removes all existing entries and adds the ones specified by *acl_entry*. To replace only a single ACL entry, see the **modify** subcommand.
pu[rge]	Purges all masked permissions. This option is useful only for ACLs that contain an entry of type **mask_obj**. Use it to prevent unintentionally granting permissions to an existing entry when a new mask is calculated as a result of adding or modifying an ACL entry.
p[ermissions]	Lists the available permission tokens.
t[est_access] *permissions...*	Tests whether or not the permissions specified in the command are granted to the principal under whose DCE identity the **acl_edit** command was invoked. This subcommand returns `Granted` if the permissions are granted, or `Denied` if the permissions are denied.
h[elp] [*command*...]	Initiates the **help** facility. If you enter only the command **h**, **acl_edit** displays a list of all commands and their functions. If you enter **h** and a command (or commands separated by a space), **acl_edit** displays help information on the specified commands.

4.4 Displaying ACL Entries

The **acl_edit p[ermissions]** and **l[ist]** subcommands display the permissions available for an object and all entries in the object's ACL. The following subsections provide information about how to use these subcommands.

4.4.1 Displaying Permissions

The exact permissions you enter for an ACL entry depend on the permissions implemented for the object by the object's ACL Manager. The ACL Manager provides the object's permissions and the meanings of those permissions to the **acl_edit** command. The **acl_edit** command can display the permissions appropriate for the object whose ACL you are editing, and it can discriminate between permissions that are valid and not valid for the object. To display an object's permission tokens and meanings, use the following **acl_edit p[ermissions]** subcommand in interactive mode:

```
sec_acl_edit> p
```

A sample of the output of the **p** subcommand follows. Remember that the permissions you see when you use the **p** subcommand differ, depending on the permission set supported by a particular ACL Manager.

```
Token   Description
r       read
w       write
x       execute
i       insert
d       delete
t       test
```

4.4.2 Displaying Entries

To display an object's ACL, use the following **acl_edit l[ist]** subcommand in interactive mode:

```
sec_acl_edit> l
```

The **l[ist]** subcommand displays the ACL entries for the object specified by *pathname* when **acl_edit** was invoked. A sample display produced by this command follows:

```
# SEC_ACL for /.../dresden.com/music/printers
# Default cell = /.../dresden.com
mask_obj:crwx---
unauthenticated:-r-----
user_obj:crwx---
user:britten:crwx---
user:mahler:-rwx---
foreign_user:/.../ud.edu/pro/bach:crwxidt #effective:crwx---
group_obj:-rwx---
group:dds:-rwx---
foreign_group:/.../china.com/writers/novelists:-r-x---
other_obj:-rwx---
foreign_other:/.../china.com:-rwx---
any_other:-r-----
extended:c417faf8-8e40-11c9-ace3-08001e55.a.b.c.a1.4.0a0b0c0d:-rwx---
```

The first line of the display shows the pathname of the object whose ACL you are editing. The default cell is listed next, and then the ACL entries. Notice that permissions not given to a principal are indicated by - (dashes) in the **acl_edit** display.

Notice also that, if a mask restricts permissions explicitly granted in the ACL entry, the resulting permissions are listed to the right of the ACL labeled as `#effective`. In the display, the entry for **foreign_user: bach** is given explicit permissions of **crwxidt**, but the ACL entry is masked by the **mask_obj** mask. Because only permissions granted by the entry and by the mask are allowed, the permissions for the foreign user **bach** are **crwx**, not **crwxidt**.

To list an object's ACLs in command-line mode, use the **-l** subcommand as shown in the following command for the object named **/my_filesystem/opus**:

```
% acl_edit /.../dresden.com/my_filesystem/opus -l
```

4.5 Adding and Modifying ACL Entries

Each ACL can contain only one entry of the same specificity. This means that there can be only one entry per principal or group for entry types of **user**, **group**, **foreign_user**, and **foreign_group**, and only one entry for all other entry types. For example, although an object's ACL can have multiple entries of type **user**, it can have only one **user** entry for a principal named **bach**. The **acl_edit** subcommands that add and modify ACL entries enforce this restriction. The commands overwrite existing ACL entries so that multiple entries of the same specificity cannot occur.

You can add the permissions in an ACL entry in any order, and you can add the ACL entries themselves in any order. Although the **list** subcommand shows dashes in place of permissions not granted by an entry, you do not need to supply dashes when you create the entry. (You can use dashes to deny access as described in Section 4.11.)

The three **acl_edit** subcommands to add and modify ACL entries are as follows:

- The **modify** subcommand adds or modifies a single ACL entry. If the entry you are modifying does not exist, **modify** adds it. If the entry you are modifying exists, **modify** replaces it.
- The **assign** subcommand adds all the entries contained in the file whose name you specify. This subcommand overwrites all existing entries in the ACL with the ones in the named file; no previous entries remain.
- The **substitute** subcommand adds or modifies all entries. Specify each entry on the command line, separating entries with a space. Like the **assign** subcommand, all existing entries are overwritten with the ones specified in the command line.

For information about performing the preceding operations in command-line mode, see Section 4.5.4.

4.5.1 Using the modify Subcommand

The following example illustrates the use of the **modify** subcommand to change an ACL entry, while operating in interactive mode.

Assume the ACL for the file **opus** has the following ACL entries:

```
mask_obj:crwxid-
unauthenticated:-r-----
user_obj:crwxid-
user:bach:crwx---
user:mozart:-r-----
group_obj:crwx---
other_obj:-r-----
```

Then assume you run the following **modify** subcommand (abbreviated as **m**):

```
sec_acl_edit> m user:mozart:crwx
```

The **modify** subcommand changes only the entry for **user:mozart** as follows:

```
mask_obj:crwxid-
unauthenticated:-r-----
user_obj:crwxid-
user:bach:crwx---
user:mozart:crwx---
group_obj:crwx---
other_obj:-r-----
```

4.5.2 Using the assign Subcommand

To use the **assign** subcommand for adding entries to an ACL, while in interactive mode, first create a file containing the ACL entries in standard

ACL entry format, with each entry on a separate line. Then execute the **assign** subcommand, specifying the name of the file of ACL, in the format

as[sign] *filename*

For example, assume the file **std_acl** contains the following entries:

```
mask_obj:crwxid-
user_obj:crwxid-
group_obj:crwx---
other_obj:-r-----
user:lizt:crwx---
group:composers:-r-----
user:bach:crwx---
user:mozart:crwx---
```

The following **assign** command, abbreviated as **as**, adds the entries in **std_acl** to the file **opus** (shown in Section 4.5.1):

```
sec_acl_edit> as std_acl
```

Because the **assign** subcommand overwrites all ACL entries with the ones on the file **std_acl**, the ACLs for the file **opus** now look like

```
mask_obj:crwxid-
user_obj:crwxid-
user:lizt:crwx---
user:bach:crwx---
user:mozart:crwx---
group_obj:crwx---
group:composers:-r-----
other_obj:-r-----
```

4.5.3 Using the substitute Subcommand

Assume you are in interactive mode and want to change all the ACL entries for a file. As an example, run the following **substitute** subcommand, abbreviated as **su**, on the ACL for the file **opus** (shown in Section 4.5.1):

```
sec_acl_edit> su user:mozart:crwx
```

The complete ACL (that is, all entries in the ACL) for **opus** now looks like

```
user:mozart:crwx---
```

4.5.4 Adding and Modifying ACL Entries in Command-Line Mode

To add or modify ACL entries in command-line mode, use the **-m** (modify), **-f** (assign), or **-s** (substitute) subcommands. For example:

- To use the **-m** subcommand to add permissions for **user:mozart** to the ACL for the file **opus**, enter the following:

```
% acl_edit /.../dresden.com/my_filesystem/opus -m user:mozart:crwx
```

  Note that you can add more than one entry at a time by using the **-m** subcommand. To do so, separate each entry with a space. For example, to add entries for user **mozart** and user **bach**, enter the following:

```
% acl_edit /.../dresden.com/my_filesystem/opus -m user:mozart:crw user:bach:r
```

- To use the **-f** subcommand to replace the entries in the ACL for the file named **opus** with the entries in **std_acl**, enter the following:

```
% acl_edit /.../dresden.com/my_filesystem/opus -f std_acl
```

- To use the **-s** subcommand to replace all entries with the one or ones specified on the command line, enter the following:

```
% acl_edit /.../dresden.com/my_filesystem/opus -s user:mozart:rw user:bach:crw
```

This example replaces all entries with the two specified by the **-s** subcommand.

4.6 Deleting ACL Entries

You can delete ACL entries by using the **acl_edit** subcommands for the mode you choose to operate in.

The following **acl_edit** subcommands let you delete ACL entries in interactive mode:

- **delete**
- **kill_entries**

The following **acl_edit** subcommands let you delete entries in command-line mode:

- **-d**
- **-k**

4.6.1 Using delete and kill_entries

Use the **delete** and **kill_entries** subcommands when working in interactive mode. For example, the following **delete** subcommand (abbreviated as **d**) deletes the **user:mozart** entry:

```
sec_acl_edit> d user:mozart
```

The following **kill_entries** subcommand (abbreviated as **k**) deletes all entries, except the **user_obj** entry if it exists:

```
sec_acl_edit> k
```

4.6.2 Deleting ACL Entries in Command-Line Mode

To delete ACL entries in command-line mode, use the **-d** (delete) or **-k** (kill_entries) subcommands. For example:

- Use the **-d** subcommand to delete the **user:mozart** entry from the ACL for the file **opus**:

  ```
  % acl_edit /.../dresden.com/my_filesystem/opus -d user:mozart
  ```

  Notice that only the type (**user**) and key (**mozart**) are required to specify the entry to delete. You do not need to enter the entire entry.

- Use the **-k** subcommand to delete all entries, except the **user_obj** entry if it is present, from the ACL for the file **opus**:

  ```
  % acl_edit /.../dresden.com/my_filesystem/opus -k
  ```

4.7 Using Masks

There are two kinds of masks that can be used on ACLs:

- The **mask_obj** mask filters permissions for all ACL entries except the **user_obj** and **other_obj** ACL entries.
- The unauthenticated mask filters permissions for principals who are not authenticated by the security server (that is, those principals who did not obtain their privilege attributes from the DCE Authentication Service).

The masks limit permissions specified in the entry to those specified in the mask. The absence of the **mask_obj** mask places no limits on the access that can be granted to a principal. The absence of the unauthenticated mask, however, prohibits any access by unauthenticated principals.

4.7.1 The mask_obj Mask

To use the **mask_obj** mask, you need to create an entry that defines the mask permissions in the form

mask_obj:*permissions*

For example, to define the **mask_obj** mask permissions as **crwx**, the ACL entry would be

mask_obj:crwx

The preceding mask permissions now filter permissions for all ACL entries except the **user_obj** and **other_obj** ACL entries.

4.7.2 The Unauthenticated Mask

To create the unauthenticated mask, you need to create an entry that defines the mask permissions in the form

unauthenticated:*permissions*

For example, to define the unauthenticated mask permissions as **crwx**, the ACL entry would be

unauthenticated:crwx

If you do not create an unauthenticated mask, the ACL Manager denies access to all principals who are not authenticated. If the mask is in place, the entries for any unauthenticated principals are filtered through the mask.

4.8 Copying ACLs

To copy an ACL from one object to another, invoke **acl_edit** in command-line mode, specifying as *pathname* the object whose ACL you want to copy. Then use the **-l** subcommand to list the object's ACL entries, and redirect the output to a file. The command to accomplish this, sending the output to the file **/tmp/tmp_acl**, is

% **acl_edit** *pathname* **-l > /tmp/tmp_acl**

Then, invoke **acl_edit** again in command-line mode, specifying as *pathname* the object that you want to copy the ACL to. Use the **-f** subcommand to replace the object's existing ACL entries with the entries saved in the file to which you redirected the output of the **-l** command. The command to accomplish this is

% **acl_edit** *pathname* **-f /tmp/tmp_acl**

You could also create a shell script to copy the ACL. The script contains the command-line commands. In addition, the script expects the pathname of the object from which the ACL is being copied to be the first argument, and the pathname of the object to which the ACL is being copied as the second argument. The script would look like

```
acl_edit $1 -l > /tmp/tmp_acl
acl_edit $2 -f /tmp/tmp_acl
```

4.9 Editing Types of ACLs

There are three basic types of DCE ACLs, which are described in detail in the following subsections. One of the three ACL types, the Object ACL, controls access to the object itself. The other two, the Initial Container ACL and the Initial Object ACL, are used as defaults to determine the protection assigned to newly created objects.

Note: Do not confuse ACL entry types (described earlier in this chapter) with ACL types described here. ACL entry types help define the purpose of specific ACL entries. ACL types define the purpose of the ACL itself.

You can use **acl_edit** to edit any ACL type by specifying the ACL type when you invoke **acl_edit**. To edit the Initial Container ACL, supply the **-ic** subcommand when you invoke **acl_edit** in either command-line or interactive mode. For example, in command-line mode, you could edit the Initial Container ACL as follows:

```
% acl_edit /.../dresden.com/my_filesystem/opus -ic
```

To edit the Initial Object ACL, supply the **-io** subcommand when you invoke **acl_edit** in either command-line or interactive mode. For example, in command-line mode, you could edit the Initial Object ACL as follows:

```
% acl_edit /.../dresden.com/my_filesystem/opus -io
```

If you invoke **acl_edit** without supplying either the **-ic** or **-io** subcommand, you edit the Object ACL.

4.9.1 Objects and Containers

The type of ACL used for an object depends on whether the object is a simple object or a container. Containers are objects that hold other objects. The objects they hold can themselves be either simple objects or container objects. Simple objects do not hold other objects. Although any DCE component can have objects and containers, the simplest and most common illustration is the file system. In the file system, there are files and directories. The files are simple objects, and the directories are containers. The directories can hold simple objects (files) and other containers (subdirectories).

The Object ACL is associated with simple and container objects. The Initial Container and Initial Object ACLs are associated only with container objects.

4.9.2 Initial ACLs for Objects and Containers

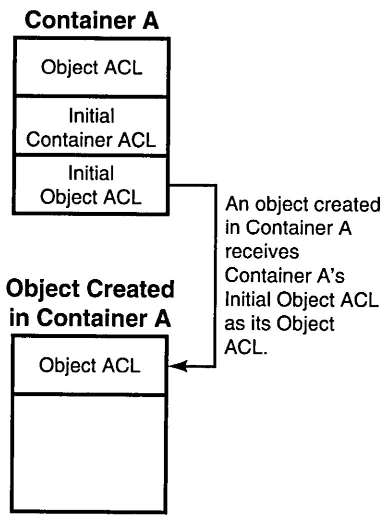
Initial ACL entries and the ACL that contains them are applied automatically when an object is created. The entries can be modified at any time with the **acl_edit** command. The types of DCE ACLs used as initial ACLs for containers and objects are as follows:

- The Initial Container ACL determines the default ACL for containers created within a container. For example, the file system Initial Container ACL for a directory specifies the default ACL for subdirectories created within that directory.
- The Initial Object ACL determines the default for objects created within a container. For example, the file system Initial Object ACL for a directory specifies the default ACL for files created within that directory.

4.9.2.1 Default ACLs for Objects

When a simple object is created in a container, it inherits the container's Initial Object ACL as its Object ACL. Figure 4-1 illustrates how the default ACL is assigned to simple objects created in containers.

Figure 4–1. Initial ACLs for Objects Created in Containers


Container A
Object ACL
Initial Container ACL
Initial Object ACL
An object created in Container A receives Container A's Initial Object ACL as its Object ACL.
Object Created in Container A
Object ACL

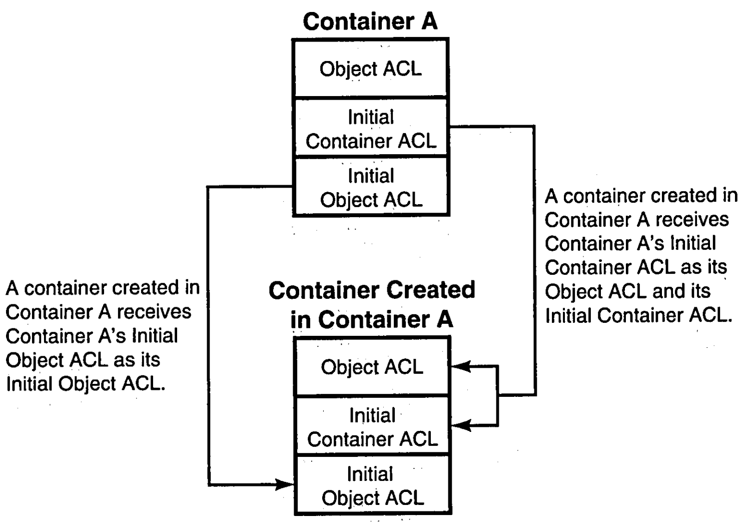
4.9.2.2 Default ACLs for Containers

When a container is created within a container (a subdirectory within a directory, for example), it inherits the parent container's

- Initial Container ACL as its Object ACL and as its Initial Container ACL
- Initial Object ACL as its Initial Object ACL

For example, if you create a file named **report** in the directory **marketing**, the system assigns **report** the Initial Object ACL of the directory **marketing**. If you create a subdirectory in **marketing**, the system assigns the new subdirectory the Initial Container ACL of **marketing**. New subdirectories also receive a set of initial ACLs that match the parent directory's initial ACLs. In this example, the new subdirectory also receives **marketing**'s initial ACLs as its own ACLs. Figure 4-2 illustrates how the default ACLs are assigned to objects created in containers.

Figure 4–2. Initial ACLs for Containers Created in Containers


Container A
Object ACL
Initial Container ACL
Initial Object ACL
A container created in Container A receives Container A's Initial Container ACL as its Object ACL and its Initial Container ACL.
A container created in Container A receives Container A's Initial Object ACL as its Initial Object ACL.
Container Created in Container A
Object ACL
Initial Container ACL
Initial Object ACL

4.9.2.3 Default Container ACL Example

The following example shows how ACLs are initially assigned to containers created within containers.

Assume Container A has the following ACLs:

Object ACL

```
user_obj:crwxid
group_obj:crwxid
other_obj:r
```

Initial Container ACL

```
user_obj:crwxid
group_obj:rw
other_obj:r
```

Initial Object ACL

```
user_obj:crwxid
group_obj:r
other_obj:r
```

When Container B is created in Container A, it has the following default ACLs:

Object ACL (Container A's Initial Container ACL)

```
user_obj:crwxid
group_obj:rw
other_obj:r
```

Initial Container ACL (Container A's Initial Container ACL)

```
user_obj:crwxid
group_obj:rw
other_obj:r
```

Initial Object ACL (Container A's Initial Object ACL)

```
user_obj:crwxid
group_obj:r
other_obj:r
```

4.10 Editing Leaf Objects in the DCE Directory Service

By default, **acl_edit** fully resolves the pathname you enter when you invoke **acl_edit**. In some cases, a pathname can resolve to a leaf object in the DCE Directory Service *and* to an object in some other DCE component that supports ACLs, like the registry. In these cases, you must use the **-e** subcommand to edit the leaf object in the Directory Service.

For example, assume a print server that implements an ACL Manager to limit access to printing services has a top-level pathname (that is, catalog point or name to which the object exports its location bindings) of the following:

```
/.../dresden.com/print_server
```

Then, the component **print_server** can be both a leaf in the CDS namespace *and* the top-level directory in the print server namespace.

To edit the ACL associated with the top level of the print server, invoke **acl_edit** as follows:

```
% dceshared/bin/acl_edit /.../dresden.com/print_server
```

To edit the ACL associated with the leaf object in the Directory Service, invoke **acl_edit** with the **-e** subcommand as follows:

```
% dceshared/bin/acl_edit -e /.../dresden.com/print_server
```

4.11 Denying Access

When you create an ACL entry for a principal, you grant only the permissions you specify in the ACL entry. To deny a principal all access to an object, create an ACL entry that contains a dash in place of the permissions. For example to deny all access to user **mozart**, the entry would be

```
user:mozart:-
```

To deny access to all unauthenticated users, do not create the unauthenticated mask. If this mask is not created (ACL entry type of **unauthenticated**), only authenticated principals can access the object. The same behavior is achieved by creating an unauthenticated mask with a dash in place of the permissions. This method also has the additional advantage of illustrating graphically that unauthenticated users have no access rights.

4.12 The Checking Sequence for ACL Entries

When an ACL Manager reads through the list of ACL entries, it first looks for a match between the privilege attributes of the principal or process desiring access and the privilege attributes listed in the ACL. When the ACL Manager finds a match, it examines the permissions in the matching ACL entry and applies the **mask_obj** mask to it (unless it is an entry of type **user_obj** or **other_obj**) if a **mask_obj** mask exists. Finally, the ACL Manager applies the unauthenticated mask if the principal is not authenticated. If the permissions that result grant the requested access, the Manager grants it to the principal. If not, access is denied.

Because the ACL Manager stops checking the ACL entries when it finds a match, it is important to understand the order in which the ACLs are checked. The ACL Managers check entries in the following order:

1. First, the ACL Manager checks the user entries, in the following order:

   a. **user_obj**

   b. **user**

   c. **foreign_user**

   The ACL Manager stops all entry checking at the first matching user entry it finds and applies the permissions in the entry. The user entries are checked in order as shown in the previous list from most specific to least specific.

2. If the ACL Manager does not find a match in the user entries, it checks *all* of the following group entries:

   a. **group_obj**

   b. **group**

   c. **foreign_group**

   If any group entries match the project list, and the logical OR of permissions from these entries does not grant access, then access is denied and no further checking is performed. Because principals accrue permissions from all groups listed in the ACL of which they are a member, *all* the groups are checked and *all* the principal's group permissions are logically ORed. This means that all permissions in all the principal's groups will be granted (unless they are restricted by a mask). The order of group entry checking is not important.

3. If the ACL Manager does not find a match between the principal requesting permission and a member of a group in the group entries, it checks the **other_obj** entry. If the ACL Manager finds a match, it stops checking ACL entries.
4. If the ACL Manager does not find a match between the principal requesting permission and the **other_obj** entry, it checks the **foreign_other** entries. If the ACL Manager finds a match, it stops checking ACL entries.
5. If the ACL Manager does not find a match between the principal requesting permission and the **foreign_other** entries, it checks the **any_other** entry. If it does not find a match in the **any_other** entry, it denies all access to the object.

4.12.1 The mask_obj Mask and ACL Checking

When the ACL Manager finds an entry that matches a principal's privilege attributes, it looks at the permissions in the entry (or, in the case of groups, entries) and at the permissions in the **mask_obj** mask, if one exists. Before it grants any permissions, it filters the entry permissions through the **mask_obj** mask. Only those permissions named in the ACL entry and in the mask are granted. For example, if an ACL entry grants **rwx** permissions and the **mask_obj** entry specifies only **r** and **w** permission, only **r** and **w** are granted. The **x** permission named in the ACL entry is ignored.

4.12.2 The Unauthenticated Mask and ACL Checking

If an ACL Manager receives an access request from an unauthenticated principal, it checks the ACL entries and applies the **mask_obj** mask, if available, as described previously. It then filters those permissions through the mask for unauthenticated principals (entry type of **unauthenticated**).

Only those permissions specified in the unauthenticated mask and in the ACL entry and **mask_obj** mask, if it exists, are granted.

4.12.3 The Effect of the Checking Order on Granting Permissions

You can think of the order in which the ACL entries are checked as going from most specific to least specific. For example, assume an ACL contains the following entries:

```
user:mahler:r
group:composers:rwx
```

If the principal named **mahler**, who is a member of the group **composers**, requests execute (**x**) access, it is denied. This happens because the order of checking specifies that all user entries (**user_obj**, **user**, and **foreign_user**) are checked before all group (**group_obj**, **group**, and **foreign_group**) entries. Therefore, the first match found by the ACL Manager is the match between user **mahler** and the ACL entry for user **mahler**. Once a matching user entry is found, checking stops and the found permissions are applied. In this case, checking stops before the **group** entry, the entry with the more liberal permissions.

4.12.4 The Effects of Denying Access

Access can be denied in user entries, as described in Section 4.11, by supplying a single dash in place of permissions in an ACL entry. If the ACL Manager finds an entry for a principal that denies access, all subsequent ACL checking stops and all access is denied.

Part 1C

DCE Distributed File Service

Chapter 5

Introduction to the DCE Distributed File Service

A distributed computing environment involves many different systems working together to perform the same types of functions usually performed on one machine in a traditional computing environment. For example, in a distributed computing environment, one system (or component) can be responsible for login procedures and file protections; another can be responsible for providing a uniform way to name and view all objects in the environment. These components communicate via a computer network, providing system access to machines and users at different geographical locations.

A distributed file system joins the file systems of individual machines into a single, global file system (filespace) accessible to many machines. OSF's distributed computing environment employs the Distributed File Service (DFS). Working with DFS is similar to working with the file system of your local machine. However, in DFS, you can access files stored on different machines in the computer network as easily as you can access files stored on your machine's local disk. You do not need to know the physical location of a file, even though a given file can be stored on any machine in the network. Simply request a file by its DCE pathname, and DFS finds the correct file automatically.

Groups of DFS users from different locations can share information and work together on the same files. By using DCE ACLs, you can easily prohibit certain users from accessing specific files or directories. This is one aspect of DFS that provides a secure computing environment.

The information in this chapter is intended as an overview of DFS. It presents concepts you need to know to use DFS efficiently and easily. The final section of the chapter includes examples of the DFS command structure and an explanation of how to get help when using DFS.

The examples in this documentation all use the DCE Directory Service naming conventions. More specifically, all examples appear in Domain Name System (DNS) format, which can differ from the format used at your site. If you have questions about the proper pathname for an object in your cell, see Part 1A of this book or consult your system administrator.

5.1 Basic Concepts

The following subsections introduce basic DFS concepts. They do not provide detailed descriptions of commands, file protection, or general conventions employed by other operating systems. Consult the documentation provided with your local operating system for information regarding that operating system.

5.1.1 The DFS Model

DFS is based on a distributed client/server model. In a client/server model, server machines store data that client machines can access. 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.

5.1.1.1 DFS Server Machines

DFS server machines run DFS server processes that provide services such as making data available and, on an administrative level, monitoring and controlling other processes. Server machines are categorized by the processes they run. For example, a server machine that runs the processes necessary for storing and sharing data assumes the role of a File Server machine. A File Server machine's processes include the Fileset Server, which provides an interface to the DFS commands and components used to manipulate filesets, and the File Exporter, which fills requests for data from client machines anywhere in the network.

Other server machines fill different roles. For example, Fileset Database machines store copies of the database that maintains the locations of system and user files, and Backup Database machines store copies of the database that contains information used to back up and restore system and user files.

5.1.1.2 DFS Client Machines

DFS client machines, generally workstations, provide computational power, access to DFS files, and other general-purpose tools to the individuals who use them. A process called the Cache Manager runs on each client machine. The Cache Manager requests data for users from the processes running on File Server machines.

When the Cache Manager receives requested data from a File Server machine, it stores copies of the data in a local storage area called the cache. The cache is an area reserved for data storage on the local disk or in memory on the client machine. The Cache Manager uses the local copy of the cached file; it does not continue to send network requests to File Server machines for data it has stored locally. Not only is access to locally stored data much quicker than access to data stored across the network, but even if the File Server machine housing the original version of the data becomes unavailable, the local copy of the data remains in the cache until the File Server machine returns to service.

As you save changes you make to data, the Cache Manager periodically sends the changed data back to the appropriate File Server machine, where your changed version replaces the data stored on the server. When the central version of the file stored on the File Server machine changes, DFS advises all other Cache Managers with copies of the file that their versions are no longer current. When other users access the file, their Cache Managers use the newer version of the data.

Some machines can act as both clients and servers. In most cases, however, you use a client machine to access files stored on a File Server machine. You can access the data after it is cached locally on the client machine.

5.1.2 Cells

In DCE, the cell is the basic unit of operation. A cell can consist of one to several thousand systems sharing a unified working environment (namespace). It is a set of server machines (File Server machines and other server machines that run special processes) and client machines that share common administration. Each cell is administered independently of other cells; its system administrators determine how its servers and workstations are configured and how much storage space is available to each user.

A cell can consist of a company, an organization, a university department, or any defined group of users. Server and client machines can be located at different geographical locations and still be members of the same cell. However, a machine can belong to only one cell at one time.

DFS presents DCE with a global filespace, independent of machine boundaries. Each cell can connect with one or more other cells running DFS (or other file systems) to provide access to an enormous amount of data. Users from different cells can share files, regardless of where the cells are located. With DFS, sharing files across the country is no different from sharing files across the building.

Although you can have access to several cells, the cell in which you are identified as an authenticated user is called your ``local cell'' (it is also sometimes referred to as your ``home cell''). All other cells to which you have access in DCE are considered ``foreign cells.'' The system administrator who configures your cell determines whether your cell participates in the global naming service. If it does, you can permit users from foreign cells that also participate in the global naming service and with which your cell shares mutual trust to access your data, and vice versa.

5.1.3 DFS Administrative Domains

DFS further simplifies the administration of a DCE cell by providing DFS administrative domains. An administrative domain is a collection of machines from the same cell configured for administration as a single unit. An administrative domain, like a cell, includes server machines that perform specific roles and client machines that access the server machines. Domains can share machines, but all of the machines in a domain must be members of the same cell.

A cell can contain a large number of machines; administrative domains can be used to simplify the administration of a large cell by organizing its machines into smaller administrative units. From a user's perspective, domains are transparent. Users who access machines and data from different domains in a common cell are still identified with respect to that same local cell.

5.1.4 DFS Administrative Lists

The administration of DFS in a cell or domain is regulated by DFS administrative lists. An administrative list is a file that determines which system administrators are allowed to issue commands that affect DFS server processes on server machines. Each process is associated with an administrative list. Individual users can be placed on an administrative list to grant them the privileges associated with the list.

Groups of users can also be created and placed on administrative lists. A group is a defined collection of users placed on administrative lists to grant the administrative privileges associated with the lists to all of the members of the group simultaneously. The same group can be included on multiple administrative lists; for a user to be extended the privileges associated with each of those lists, the user needs only to be added to the proper group. This feature greatly simplifies the responsibilities of the DFS system administrator.

For example, the Fileset Server on each File Server machine has associated with it an administrative list named **admin.ft**. Each user and each member of a group listed in the **admin.ft** file on a machine can issue administrative-level commands that affect the Fileset Server on that machine. A user can acquire the administrative privileges associated with the **admin.ft** file either directly, by being added to the file, or indirectly, by being added to a group that is listed in the file.

Topics related to DFS system administration are described in detail in the *OSF DCE Administration Guide*.

5.1.5 DFS Security

Several security measures are used to ensure that only valid users can access files stored in the DFS filespace. When you correctly authenticate to a DCE cell, you receive authentication information in the form of a ticket. Your ticket acts as proof to DFS File Server machines that you are an authenticated user and can access data in the filespace.

When you attempt to access data via a client machine, the Cache Manager presents your ticket to the File Server machine that houses the data. The Cache Manager receives the data you requested when mutual authentication is complete between the Cache Manager and the File Server machine. Mutual authentication is achieved when the two machines prove their identities to each other. DFS requires this mutual authentication whenever a server and a client communicate.

With DCE ACLs, you designate who can access the information in your files. An ACL can exist for each directory and each file in DFS, specifying the actions that different users can perform on the directory or file. (ACLs exist only for DCE Local File System directories and files; see the following section for more information about the DCE Local File System.) An individual with the necessary permissions on a file or directory can determine the users who appear on its ACL and the permissions they have. Groups of users can also be defined and added to ACLs just as individual users are added.

Refer to Chapter 7 for more information about using ACLs to protect files in DFS; refer to Part 1B of this book for further information about ACLs.

5.2 DFS Components and Features

Many elements of DFS are virtually transparent to system users. System administrators, however, rely heavily on these components to provide the increased efficiency and better system reliability mentioned previously. The following subsections describe some of the components and concepts unique to DFS. They are intended only as overviews; detailed information about these topics is provided in the *OSF DCE Administration Guide*.

5.2.1 The DCE Local File System

DFS provides a high-performance, log-based file system: the DCE Local File System (DCE LFS). DCE LFS enhances the performance and reliability of your local file system by improving the way data is stored and managed. In addition to DCE LFS, DFS also supports the use of other file systems. However, other file systems do not support many of the benefits available with DCE LFS.

From a user perspective, the primary storage elements of DCE LFS are filesets and aggregates. In a file system hierarchy, your files are contained in directories, the directories are located on filesets, and the filesets reside on aggregates. Filesets and aggregates are analogous to the file systems and disk partitions used in other file systems; to the user, there is little difference.

Most file systems store data on disk partitions, which are segments of the physical disk. DCE LFS stores filesets on aggregates, which are also segments of the physical disk. Both partitions and aggregates are housed on workstations or File Server machines, where the data they contain is accessed by client machines for users of the system.

However, while a DCE LFS aggregate is physically equivalent to a disk partition, it also contains metadata about the structure and location of information it contains. In addition, a DCE LFS aggregate contains a log of all operations, such as file creation and modification, that affect the metadata.

The DCE LFS log provides additional protection for your data. In the event of an abnormal system shutdown, DFS replays the logged information about the metadata and uses it to return the aggregate to a consistent state. Like many features of DCE LFS, the log is completely transparent to users.

5.2.2 Filesets

DCE LFS aggregates also support the use of DCE LFS filesets. A fileset is a hierarchical collection of files managed as a single unit. DCE LFS filesets can vary in size, but they are almost always smaller than an aggregate (they can never be larger). As a result, multiple DCE LFS filesets can be stored on a single aggregate. This allows system administrators to maximize machine and disk usage.

By comparison, a non-LFS fileset is equal in size to a non-LFS partition (for example, a UNIX file system is equal in size to a UNIX partition). A non-LFS partition can house only a single file system (non-LFS fileset). (Section 5.4 briefly discusses some important differences between DFS and other file systems.)

DCE LFS filesets are also easier to manage than non-LFS filesets. System administrators can easily move DCE LFS filesets from one aggregate to another or from one machine to another. This allows administrators to balance the load on the system across the available machines, generally improving access time to data for all users. As a DFS user, you need never know the current location of your data because the Cache Manager on the client machine you are using contacts the Fileset Location Server (FL Server), the DFS server process responsible for fileset location management, to determine the location of your data.

All of a user's files and directories are usually stored on a single fileset. The fileset resides on a single aggregate (or partition) and File Server machine, where it is managed as a single unit, separate from the data of other users.

5.2.2.1 Fileset Identification

In DCE LFS, each fileset has a name unique to the cell in which it is stored. This is true of both DCE LFS filesets and non-LFS filesets (filesets that reside on non-LFS partitions instead of aggregates). User filesets are typically named **user.***username*, where *username* is the login name of the user whose data resides on the fileset. For example, the fileset that stores the data for a user whose login name is **terry** has the name **user.terry**. When you access **terry**'s data, you are actually accessing the fileset named **user.terry**, which resides on some File Server machine in **terry**'s local cell.

Each fileset is also associated with a single, unique fileset ID number. Each fileset ID is displayed as two numbers separated by ,, (a pair of commas). For example, the fileset named **user.terry** may have the fileset ID number **0,,12262**. When using DFS commands, you can usually reference filesets by name or by ID number. When using a fileset ID number in a command, you can omit the first digit and both commas if the first digit is a 0 (zero). More information on using fileset names and ID numbers appears in Chapter 6.

5.2.2.2 Fileset Quota

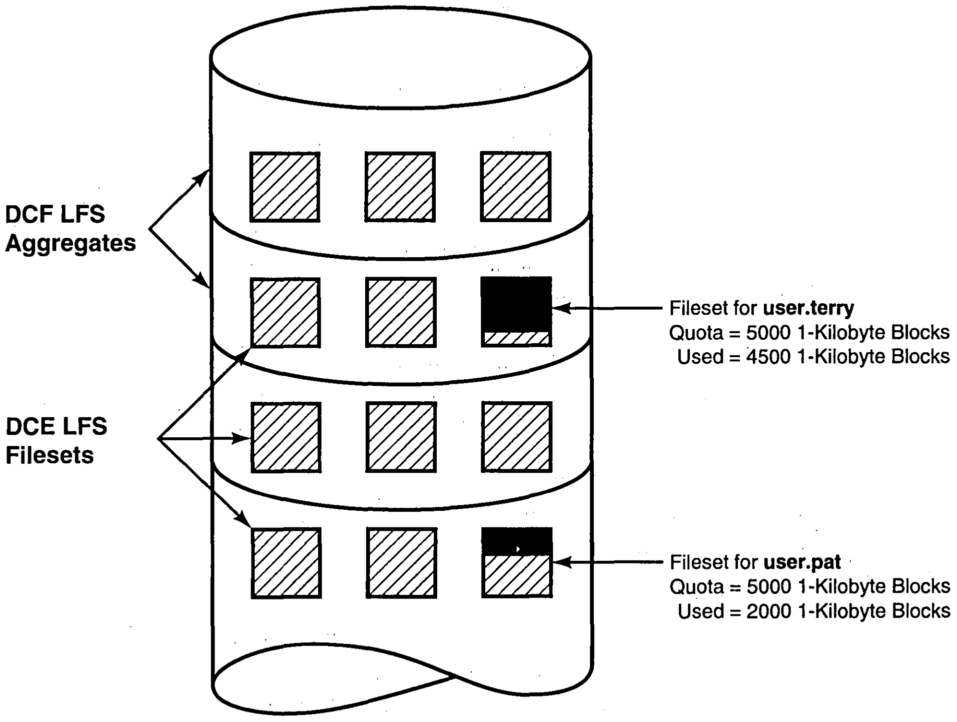
Every fileset, both DCE LFS and non-LFS, has a fileset quota, or size limit, assigned by the system administrator. A fileset's quota determines the maximum amount of data that can be stored on the fileset. For DCE LFS filesets, the quota defines the amount of disk space allocated to the fileset on the 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 5000 kilobytes. A system administrator can change the quota of a DCE LFS fileset; the quota of a non-LFS fileset always equals the size of the partition on which it resides.

Different filesets can have different quotas. If you wish to copy files from a directory on your fileset to a directory on another user's fileset, remember that the other user's fileset may have a smaller quota than yours and may not have enough space to accommodate the file.

Figure 5-1 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, **terry** and **pat**. Both filesets have a quota of 5000 1-kilobyte blocks, the default associated with every new DCE LFS fileset. However, **terry**'s fileset, **user.terry**, already contains 4500 1-kilobyte blocks of data, while **pat**'s fileset, **user.pat**, contains only 2000 1-kilobyte blocks of data.

Figure 5–1. Fileset Quotas


DCF LFS Aggregates
DCE LFS Filesets
Fileset for **user.terry**
Quota = 5000 1-Kilobyte Blocks
Used = 4500 1-Kilobyte Blocks
Fileset for **user.pat**
Quota = 5000 1-Kilobyte Blocks
Used = 2000 1-Kilobyte Blocks

A 750-kilobyte file can be copied to **user.pat** because that fileset has enough quota remaining to accommodate 750 kilobytes of data. However, the same file cannot be copied to **user.terry** because that fileset has only 500 1-kilobyte blocks of quota remaining. Chapter 6 explains how to check the quota of a DCE LFS or non-LFS fileset.

5.2.2.3 Fileset Accessibility

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

In DCE LFS, each fileset has a working, or 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, but it can be read and executed.

The replication of commonly used files, such as those for text editors, greatly reduces the chance that heavily used applications are unavailable. With replication, the unavailability of a single server machine housing a replicated DCE LFS fileset does not interrupt work involving that fileset because copies of the fileset are still available on other machines. For example, if a machine that houses the file for an editor you are using becomes unavailable, the editor is still available to you from another machine.

Replication also prevents one machine from becoming overburdened with requests for popular files from a DCE LFS fileset. Different users can be given access to copies of application files stored on different machines to reduce the number of users accessing any one machine. Because they are seldom in demand by more than one user, user filesets are rarely replicated.

Backup DCE LFS filesets are copies of the files in a read/write DCE LFS fileset made at specific times. A backup fileset saves the versions of the files in the original fileset as they existed at the time the backup was made. Backup filesets are typically available from directories with names like **.OldFiles** or **.BackUp**. They are usually made once a day, at times of low system usage, such as very early in the morning.

A backup fileset does not reflect any changes you made to data in the original fileset since the backup was created. But it does allow you to recapture a fairly recent version of your data without assistance from a system administrator. For example, if you accidentally delete a file or directory, you can copy the data from the backup fileset to your working (read/write) fileset to access the most recently backed-up version. Note that, like the data in read-only filesets, the data in backup filesets cannot be changed. To edit data in the backup version of a fileset, you must first copy the data to a regular directory in a working fileset.

Your system administrator is responsible for the creation of backup DCE LFS filesets. Consult your system administrator to determine if backup filesets are used in your cell.

DFS also includes the DFS Backup System, which supports the backing up of all filesets, both DCE LFS and non-LFS, to tape. Once data from filesets is copied to tape, it can be restored to the file system in the event of data loss. Consult your system administrator to determine how often the filesets in your cell are backed up to tape.

5.2.3 Mount Points

DFS translates a directory name into a fileset name and automatically tracks every fileset's location, even when the fileset is moved between aggregates or between machines. You never need to remember the machine locations of your files.

In DFS, access to a fileset is provided through a mount point. A mount point indicates the storage location of the fileset and enables you to access files stored on different filesets, aggregates, or File Server machines. A mount point looks and acts like a normal directory; any time you change from one directory to another, you can cross one or more mount points along the way.

For example, when you access data on the fileset named **user.terry**, the Cache Manager on the client machine to which you are connected encounters the mount point for the fileset. The mount point directs the Cache Manager to the File Server machine and aggregate on which the fileset is stored.

Because different filesets can be stored on different File Server machines, if one machine is unavailable, only the filesets housed on that machine are inaccessible; files stored on other File Server machines are still available. However, you cannot access a directory if it crosses a mount point on an inaccessible machine.

5.3 Interaction with Other DCE Components

DFS interacts at some level with all of the other DCE components. From a user perspective, most of the other components do not directly impact your use of DFS. However, to use DFS most effectively, you need to know how the system interacts with the DCE Security Service and the DCE Directory Service.

DFS uses the following components of the DCE Security Service to manage security and data protection:

- The DCE Authentication Service ensures that only authorized users can use the system, and it ensures that only authorized machines can communicate with other machines in the network.
- The DCE Registry Service maintains the registry database, which contains user, group, machine, and account information. (Accounts determine who can log into the system, and they maintain information such as user passwords and home directories.)
- The DCE Privilege Service verifies that users have the necessary permissions to perform operations they request.
- The DCE Access Control List (ACL) Facility provides an interface that allows you to set protections on DCE LFS file system objects (files and directories).
- The DCE Login Facility initializes a user's security environment in DCE, gathering authentication information associated with the user for use with other services (such as DFS).

Refer to Part 1B of this book for more information about the DCE Security Service; see Chapter 7 for more information about using DCE ACLs in DFS.

DFS uses the following components of the DCE Directory Service to name and locate objects:

- The DCE Cell Directory Service (CDS) manages names within a cell.
- The DCE Global Directory Service (GDS) supports the X.500 global naming environment between cells and outside of cells.
- The DCE Global Directory Agent (GDA) allows CDS to participate in the global naming environment via either GDS or the Domain Name System (DNS), a widely used global naming service.

Refer to Part 1A of this book for more information about the DCE Directory Service; see Chapter 6 for more information about object naming conventions in DFS.

5.4 DFS and Your Native File System

As described previously, some differences exist between DFS and the file system employed by your local operating system. Most of these differences are transparent at a user level. For example, basic file manipulation commands (such as the **cd** and **ls** commands used in the UNIX operating system) are the same in DFS and your local file system. The following paragraphs describe some of the procedural differences that you may notice when using DFS with your operating system.

In a nondistributed file system, when accessing a file on a remote machine, you must either log into the remote machine or explicitly transfer the file from the remote machine to the local machine. In DFS, when accessing a file on a remote machine, you need only specify the pathname of the file. File sharing in DFS is not restricted by operating system or machine differences. A DFS user with the necessary permissions can access a file on any machine in any cell just by specifying the full pathname of the file.

When using a nondistributed file system, if your local machine becomes unavailable, the entire file system becomes inaccessible. When using DFS, a File Server machine outage means either you cannot access certain files or you cannot make permanent changes to files located on that File Server machine. Normally the whole system does not become unavailable, however, because your system administrator stores copies of popular application programs on multiple File Server machines. If a single File Server machine housing a copy of a replicated program becomes unavailable, you still have access to copies of the program stored on other server machines.

5.5 DFS Commands and Help

DFS commands are divided into the following categories, or command suites:

- The **bak** command suite. Commands in this suite are not issued by users; they are used by system administrators to operate the DFS Backup System.
- The **bos** command suite. Commands in this suite are issued by users to list cell information; they are used by system administrators to monitor and control DFS server processes and security.
- The **cm** command suite. Commands in this suite are issued by users to determine information about machines, files, and directories; they are used by system administrators to set **setuid** status and to alter and configure the Cache Manager.
- The **fts** command suite. Commands in this suite are issued by users to check fileset quota information; they are used by system administrators to manipulate filesets.

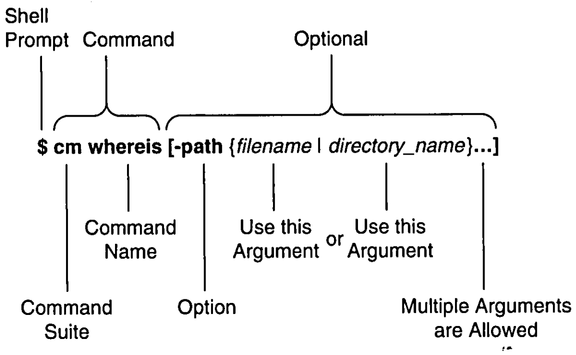
DFS commands are divided into user-level commands and administrative-level commands. All users can issue user-level commands; only administrative users can issue administrative-level commands. DFS administrative commands are discussed in detail in the *OSF DCE Administration Guide* and the *OSF DCE Administration Reference*.

5.5.1 DFS Command Structure

All DFS commands have the following structure:

command {*-option1 argument...* | *-option2* {*argument1* | *argument2*}...} [*-optional_information*]

The following example illustrates the elements of a DFS command:



The following list summarizes the elements of a DFS command:

Command	A command consists of the command suite (**cm** in the preceding example) and the command name (**whereis** in the example). The command suite and the command name must always be typed together, separated by a space. The command suite specifies the group of related commands to which the command belongs; the command name directs the server process or program to perform a specific action. Both the command suite and the command name always appear in bold font in the text.
Options	Command options always appear in bold font in the text, are always preceded by a **-** (dash), and are often followed by arguments. In the previous example, **-path** is an option, with *filename* and *directory_name* as its arguments. An option and its arguments tell the server process or program which entities to manipulate when executing the command (for example, which file or directory). In general, provide the options for a command in the order detailed in the documentation.
Arguments	Arguments for options always appear in italic font in the text. The **{ | }** (braces separated by a vertical bar) indicate that you can enter only one of two possible arguments (or use only one of two possible options). In the previous example, you can enter

either a *filename* or a *directory_name*; the ... (ellipsis) following the closing brace indicates that multiple *filename*s, *directory_name*s, or both can be entered.

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

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

5.5.1.1 Command Shortcuts

When supplying an argument (such as a *filename* or *directory_name* in the previous example), you can omit the option (such as **-path** in the example) associated with the argument if

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax. (The syntax for each command is listed with its description in Part 2 of this book or in the *OSF DCE Administration Reference*.)
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.
- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

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

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

You can also abbreviate a command name to the shortest form that still distinguishes it from the other command names in its suite. For example, you can shorten the **fts help** command to **fts h** because no other command names in the **fts** command suite begin with the letter "h." However, there are several **fts** commands that begin with the letter "l," such as **fts lsquota**, **fts lsmount**, and others. To avoid ambiguity, you can abbreviate these commands to **fts lsq** and **fts lsm**; other **fts** command names that begin with "l" can be abbreviated in a similar fashion.

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

Complete command:

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

Abbreviated command name and abbreviated option:

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

Abbreviated command name and omitted option:

```
% fts lsq jlw/doc jlw/public
```

5.5.2 Receiving Help

There are several different ways to access help about DFS commands. The following list summarizes the syntax for the different help options:

- To view the introductory page for a command suite, enter **man** followed by the command suite at the system prompt, as follows:

  ```
  % man fts
  ```

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

```
% man fts_lsquota
```

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

```
% fts help
```

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

```
% fts help lsquota
```

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

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

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

```
% fts apropos quota
```

Chapter 6

Accessing Data in DFS

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

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

The information in this chapter details how to change to a different directory, how to list the contents of a directory, and how to check the locations of files and directories. In addition, it also describes how to check your fileset quota (the amount of data you can store) and demonstrates how to check the status of each File Server machine with which your client machine has been in contact. It is assumed that you understand Part 1A of this book.

6.1 Naming Conventions

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

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

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

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

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

Your system administrator determines if these prefixes are used in your cell. If they are, use them only interactively (at the command line). Do not include them in persistent storage (for example, in a shell script).

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

6.2 Accessing Files and Directories

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

In DFS, the permissions associated with a DCE LFS file or directory are set using DCE Access Control Lists (ACLs). (Permissions are set for non-LFS files and directories with the standard operating system mode bits.) DFS uses a specific implementation of DCE ACLs. See Part 1B of this book for complete details about using DCE ACLs. See Chapter 7 for information about using ACLs to limit access to files and directories in DFS.

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

- Be authenticated to DCE
- Specify correct pathnames
- Have access to the desired files and directories based on the ACLs associated with them

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

6.2.1 Changing to a Different Directory

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

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

```
% pwd

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

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

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

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

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

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

6.2.2 Listing the Contents of a Directory

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

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

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

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

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

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

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

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

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

6.2.3 Using Name Prefixes in Commands

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

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

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

% cd /:/usr/terry/public
```

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

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

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

% ls /:/usr/terry/public
```

6.3 Locating Files and Directories

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

In addition, when using DCE LFS, your system administrator can move filesets to different File Server machines to use system disk storage efficiently. Therefore, you sometimes need to check the location of your fileset to verify its location in the event of a machine outage.

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

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

6.3.1 Using the cm whereis Command

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

% **cm whereis** [**-path** {*filename* | *directory_name*}...]

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

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

% **cm whereis /.../abc.com/fs/usr/terry**

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

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

% cm whereis plans /.../abc.com/fs/usr/pat/strategy

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

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

6.4 Listing Fileset Quota Information

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

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

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

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

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

The command displays the following information about a non-LFS fileset:

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

Files stored on a non-LFS partition are still considered to be stored on a fileset. However, the fileset in this case is equal in size to the disk partition on which the fileset resides. The quota of the fileset, therefore, is equal to the size of the disk partition. (In the UNIX operating system, you can use the **df** command to check the size of the partition; in DFS, you can use the **fts lsquota** command to determine the size of the fileset.)

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

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

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

6.4.1 Using the fts lsquota Command

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

% **fts lsquota** [{**-path** {*filename* | *directory_name*}... | **-fileset** {*name* | *ID*}...}]

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

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

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

% **fts lsquota**

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

The following example displays quota and usage information about a non-LFS fileset, **user.jlw**, which contains the directory named **/.../abc.com/fs/usr/jlw**. Because **user.jlw** is a non-LFS fileset, the `Quota`, `Used`, and `% Used` columns contain 0 (zeros).

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

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

Fileset Name    Quota   Used   % Used   Aggregate
user.jlw            0      0       0%     84% = 8448/10000 (non-LFS)
```

6.5 Checking File Server Machine Status

File Server machines in your cell sometimes become unavailable due to hardware problems, software problems, or routine maintenance. If you experience problems using binary files or saving files, you can use the **cm statservers** command to check the statuses of the File Server machines housing the files. When a File Server machine is unavailable, you cannot access files stored on that machine (or save any changes you made to cached versions of files stored on that machine) until the machine returns to service.

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

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

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

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

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

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

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

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

6.5.1 Using the cm statservers Command

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

% **cm statservers** [{**-cell** *cellname* | **-all**}] [**-fast**]

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

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

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

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

% **cm statservers -all &**

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

Chapter 7

Protecting Data in DFS

This chapter provides an overview of how you can protect your data in DFS. You protect DCE LFS data by using ACLs to supplement the UNIX file system's protection of mode bits. With ACLs, you can grant or deny individual users or groups access to your DCE LFS data. This allows you to set your own security on your files and directories.

This chapter provides information about the elements of an ACL and how you can use these elements to create ACL entries for users and groups. It includes examples of the different types of entries you can create. It also describes how DFS evaluates an object's ACL to determine a user's permissions for the object. It concludes by describing the interaction between ACLs and UNIX mode bits and briefly discussing the commands and permissions necessary to set ACLs. Because ACLs are not supported for non-LFS file systems, UNIX mode bits are still used to provide protection for most non-LFS data.

DCE ACLs are used to set the permissions associated with a file or directory in a DCE LFS fileset. The information in this chapter is presented as summary information only. Refer to Part 1B of this book for complete details about creating and manipulating ACLs.

7.1 ACLs in DFS

The ACL for a directory or file object in DFS comprises multiple entries defining who is authorized to access the object and what operations they can perform on it. When a user requests that the system perform an operation on an object, the system examines the ACL for the object to determine if the user has the necessary permissions for the operation. If the user has sufficient permissions, the system completes the operation; if the user has insufficient permissions, the system does not perform the task.

An ACL entry can define the permissions granted to a user or group for an object, or it can define an ACL mask for that object. Each entry includes the following elements:

type[:*key*]:*permissions*

The subsections that follow describe the type, key, and permissions elements of an ACL entry. Subsequent sections describe how you can use the different elements of an ACL entry to set the permissions for an object.

The system administrator who creates a fileset determines the initial ACLs associated with the root directory of that fileset. These initial ACLs are then inherited and used as defaults by new objects created in the fileset. Refer to Part 1B for more information about ACL inheritance.

Note: A directory is referred to as a ''container object'' because it can contain other objects; a file is referred to as an ''object.'' The type of an object (container object or simple object) is set when it is created; it cannot be altered. This chapter uses the term ''object'' to refer to either a directory or a file. In most cases, it explicitly uses the terms ''directory'' and ''file.''

7.1.1 Entry Types

The *type* element of an ACL entry specifies whether the entry defines the permissions for a user, a group, or a mask. Table 7-1 lists the entry types for users and groups and describes the users and groups to which each type applies.

Table 7–1. ACL Types for Users and Groups

Type	**Applies to**
user_obj	The user who owns the object
user	A specific user from the local cell
foreign_user	A specific user from a specific foreign cell
group_obj	Members of the group that owns the object
group	Members of a specific group from the local cell
foreign_group	Members of a specific group from a specific foreign cell
other_obj	Users from the local cell who do not match any of the above entries
foreign_other	Users from a specific foreign cell who do not match any of the above entries
any_other	Users who do not match any of the above entries

Two additional entry types of importance are also available:

- The **mask_obj** entry type specifies the subset of permissions that can be granted to any entries except the **user_obj** and **other_obj** entries. Permissions granted to users from any of the other ACL entries are filtered through the **mask_obj** entry; only those permissions found in both a user's entry and the **mask_obj** entry are granted to the user.
- The **unauthenticated** entry type specifies the subset of permissions that can be granted to an unauthenticated user (a user whose identity cannot be verified by the DCE Security Service). Permissions granted to an unauthenticated user from an ACL entry are filtered through both the **mask_obj** entry (if applicable) and the **unauthenticated** entry; only those permissions found in an unauthenticated user's entry, the **mask_obj** entry (if applicable), and the **unauthenticated** entry are granted to the unauthenticated user.

7.1.2 Entry Keys

The *key* element of an ACL entry identifies the user or group for which the entry is defined. The name included in an ACL for a user is the user's login (principal) name; the name included in an ACL for a group is the name defined for the group in the registry database. (Although the principal names of server machines can also be included on ACLs, this chapter discusses only the inclusion of users and groups on ACLs.)

The type of user or group specified with the key must match the entry type. For example, the login name of a user must be specified with an entry of type **user**. If the entry is for a foreign user or group, the full pathname of the foreign cell must precede the name of the user or group (for example, **/.../abc.com/writers** for a group named **writers** from the cell named **abc.com**).

A key is used only with the **user**, **foreign_user**, **group**, **foreign_group**, and **foreign_other** entry types; it is omitted from all other types of entries.

7.1.3 Entry Permissions

A set of *permissions* in an ACL entry defines the operations the user or group to which the entry applies can perform on the object. For a **mask_obj** entry or an **unauthenticated** entry, the permissions specified define the subset of permissions that can be granted to any entries to which the mask applies.

The following six permissions are available for DCE ACLs on DCE LFS directory and file objects. Any combination of these permissions can be associated with an entry; a **-** (dash) is displayed in place of an omitted permission in an entry.

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

Table 7-2 lists the various operations that can be performed on a directory or file and describes the ACL permissions on objects required to perform them. (Note that the insert and delete permissions are meaningless when applied to a file.)

Table 7–2. Directory and File Operations and Required ACL Permissions

Operation	**Permissions on Objects**	
Change to a directory	**x**	All directories that lead to the directory
	x	The directory itself
List the objects in a directory	**x**	All directories that lead to the directory
	r	The directory itself
List information about the objects in a directory	**x**	All directories that lead to the directory
	rx	The directory itself
Create an object	**x**	All directories that lead to the directory in which the object is to be placed
	wxi	The directory in which the object is to be placed
Delete an object	**x**	All directories that lead to the directory from which the object is to be deleted
	wxd	The directory from which the object is to be deleted
Rename an object	**x**	All directories that lead to the object's current directory
	wxd	The object's current directory
	x	All directories that lead to the object's intended directory
	xwi	The object's intended directory

Operation	Permissions on Objects	
Read or read-lock a file	x	All directories that lead to the file
	r	The file itself
Write or write-lock a file	x	All directories that lead to the file
	w	The file itself
List the ACLs on an object	x	All directories that lead to the object
Change the ACLs on an object	x	All directories that lead to the object
	c	The object itself

Note: In Table 7-2, the operation "List the objects in a directory" refers to displaying a simple list of the objects in a directory (for example, using the UNIX **ls** command with no options or using **ls -a**). The operation "List information about the objects in a directory" refers to obtaining more detailed information about the objects in a directory, such as each object's mode bits or the time of its most recent update (for example, using the UNIX **ls -l** or **ls -t** command). The latter operation is referred to as a "lookup" of the objects in a directory.

Also, all operations require that the issuer have the execute (**x**) permission on all directories that lead to the object to be manipulated. Keep this in mind as you read the following sections in this chapter.

7.2 ACL Entries

The following subsections describe the various types of ACL entries you can specify to define the permissions granted to object owners, users, and groups. They also include information about entries for ACL masks, which you can use to restrict the permissions granted to different users and groups.

7.2.1 ACL Entries for Owners

ACL entries for owners of an object are specified with the **user_obj** and **group_obj** entry types. Only one user and one group can own an object. The user and group owners of an object are automatically assigned when the object is created. They cannot be changed with the **acl_edit** command. (They are changed with the UNIX **chown** and **chgrp** commands or their equivalents.)

The following entries allow the user and group that own an object to perform any operations on the object. Note that, because the owners of an object cannot be changed with the **acl_edit** command, the **user_obj** and **group_obj** entries do not include a key element.

```
user_obj:rwxcid
group_obj:rwxcid
```

The permissions specified for the **group_obj** entry are subject to filtering through the **mask_obj** entry. More information about masks follows in Section 7.2.4.

7.2.2 ACL Entries for Users

An ACL entry for a user specifies that user's permissions for the object with which the ACL is associated. For example, the following entries define the permissions granted to two individual users for a directory object. The first user is from the local cell; the second user is from a foreign cell.

```
user:jlw:rwx-id
foreign_user:/.../abc.com/wvh:r-x---
```

The first ACL entry is for the **user** whose login (principal) name, as specified by the key, is **jlw**. The user is from the local cell. The permissions granted to **jlw** are read, write, execute, insert, and delete (as indicated by the initials **r**, **w**, **x**, **i**, and **d**). These permissions allow **jlw** to perform the following operations:

- The **x** permission allows the user to change to the directory, list its contents, and view the ACLs of the objects it contains.

- The **r** and **x** permissions allow the user to list information about the objects in the directory.
- The **w**, **x**, and **i** permissions allow the user to insert new objects into the directory.
- The **w**, **x**, and **d** permissions allow the user to delete existing objects from the directory.

The **-** (dash) in the entry indicates a permission that is not granted. A full permission set is **rwxcid**, so the control (**c**) permission is not granted to **jlw**. As a result, the user cannot modify the ACLs associated with the directory.

The second entry is for the **foreign_user** whose login name is **wvh**. The user is from the cell named **abc.com**. The permissions granted to **wvh** are **r** and **x**, which allow the user to change to the directory, list information about its contents, and view the ACLs of the objects it contains, but nothing more.

Permissions specified with **user** and **foreign_user** entries are subject to filtering through the **mask_obj** entry. More information about masks follows in Section 7.2.4.

7.2.3 ACL Entries for Groups

An ACL entry for a group specifies the permissions granted to members of that group for the object with which the ACL is associated. The following entry specifies the permissions granted to a group for a directory object:

```
group:writers:r-x---
```

The entry type is for a **group** whose name, as specified by the key, is **writers**. The permissions granted to the group are read and execute, meaning members of the group can change to the directory, list information about its contents, and examine the ACLs of objects it contains. The **-** (dashes) indicate that the write, control, insert, and delete permissions are not granted to the group, so the members of the group can perform no additional operations on the directory.

Defining an ACL entry for a group of users enables you to control access to an object for a number of users at one time; you need not define an entry for each individual user. For example, if you have the control permission for a file named **purchasing.memo**, you can add the following pair of entries to the ACL for that file to grant the read, write, and execute permissions to each member of the **purchasing** group and the read, execute, and control permissions to each member of the **finance** group:

```
group:purchasing:rwx---
group:finance:r-xc--
```

Moreover, because a user accrues permissions from all of the groups to which the user belongs, a user who belongs to both groups is granted the **r**, **w**, **x**, and **c** permissions. A user who is a member of only one of the two groups is granted only the permissions associated with that group.

Because of the way the system evaluates ACLs, if a user has an individual entry in an ACL, the user is not granted the permissions associated with any groups to which he or she belongs. Section 7.3 provides more information about how the system evaluates ACLs. In addition, permissions specified with **group_obj**, **group**, and **foreign_group** entries are subject to filtering through the **mask_obj** entry. The following subsection provides more information about masks.

7.2.4 ACL Entries for Masks

An ACL entry of type **mask_obj** restricts the permissions granted to all entry types *except* the **user_obj** and **other_obj** types. When evaluating any entry other than a **user_obj** or **other_obj** entry, the system filters the entry's permissions through the **mask_obj**. Only those permissions found in both the user's entry and the **mask_obj** entry are granted. The following entry specifies the permissions allowed by the **mask_obj** for an object:

```
mask_obj:rwx---
```

Entry types to which this mask applies can grant only the read, write, and execute permissions for the object (the entries may grant even fewer permissions, depending on the permissions they include). For example, the following entry on the same object grants user **jlw** the **r**, **w**, and **c** permissions:

```
user:jlw:rw-c--
```

However, the **mask_obj** defined previously grants only the **r**, **w**, and **x** permissions. Therefore, the system grants **jlw** only the **r** and **w** permissions; the mask denies the user the **c** permission. In addition, because masks can only restrict, not extend, permissions for a user or group, the system does not grant the **x** permission associated with the mask to the user.

Were the above entry displayed from an actual ACL, the entry would include the label `#effective` to show the permissions that remain after it is filtered through the **mask_obj**. For example, because the previous entry retains only the **r** and **w** permissions, it would appear as follows in an actual ACL:

```
user:jlw:rw-c-- #effective:rw----
```

An ACL entry of type **unauthenticated** restricts the permissions granted to *all* unauthenticated users. A user can access DCE without being authenticated; for example, the user can log into a local machine without authenticating to DCE, or the Security Server that handles authentication may be unavailable when the user logs in. In such cases, the system cannot verify that users are who they claim to be, so users are said to be unauthenticated to DCE.

Permissions granted to an unauthenticated user from an ACL entry are filtered through the permissions specified with the **unauthenticated** entry; only those permissions found in both the entry that applies to the user and the **unauthenticated** entry are granted. (The user's permissions can be further restricted if they are also filtered through the **mask_obj** entry.)

For example, suppose the entries for user **jlw** and the **mask_obj** exist as described previously, but the **unauthenticated** entry grants only the **r** permission:

```
mask_obj:rwx---
user:jlw:rw-c--
unauthenticated:r-----
```

If user **jlw** is authenticated, the system grants **jlw** the permissions mentioned previously (**r** and **w**). However, if **jlw** accesses DCE without being authenticated, the system filters **jlw**'s permissions through the **unauthenticated** entry and grants the user only the **r** permission (the only permission common to all three entries).

Omitting the **mask_obj** entry from an ACL is equivalent to specifying it with a full permission set (**rwxcid**); a user's permissions are based solely upon the permissions granted to the user or to the groups to which the user belongs. Omitting the **unauthenticated** entry is equivalent to specifying it with no permissions; an unauthenticated user has no permissions. (Note that a **mask_obj** entry can be displayed with an ACL even if none has explicitly been defined. Such an entry represents the implicit **mask_obj** associated with the ACL.)

7.3 ACL Evaluation

When a user requests that an operation be performed on an object, the system examines the object's ACL to determine the user's permissions. The system examines the entries in an ACL in the order described in the following list. It stops checking a user's permissions as soon as the user matches a qualification described in the list. Remember as you read the list that the evaluation of a user's permissions progresses to a step only if the user fails to match all previous qualifications.

1. The user owns the object. The system grants the user the permissions specified with the **user_obj** entry. These permissions are *not* filtered through the **mask_obj**.

2. A **user** or **foreign_user** entry exists for the user. The system grants the user the permissions specified with that entry after filtering the entry through the **mask_obj**.

3. The user belongs to the group that owns the object (the owning group's permissions are specified with the **group_obj** entry) or to any other groups that have **group** or **foreign_group** entries. If the evaluation progresses to this step, the user accrues permissions from all of the groups to which he or she belongs. The system filters the permissions accrued from all of the groups to which the user belongs through the **mask_obj** and grants the user only those permissions found in both the groups and the **mask_obj**.

4. The user is from the local cell. The system grants the user the permissions specified with the **other_obj** entry. These permissions are *not* filtered through the **mask_obj**.

5. The user belongs to a foreign cell that has a **foreign_other** entry. The system grants the user the permissions specified with the entry associated with that cell after filtering the entry through the **mask_obj**.
6. The system grants the user the permissions specified with the **any_other** entry after filtering the entry through the **mask_obj**.

After the system matches the user attempting to perform an operation with one of the previous qualifications, it determines whether the user is authenticated. If the user is authenticated, the system grants the user the permissions associated with his or her authenticated identity. Otherwise, the system filters the permissions associated with the identity through the unauthenticated mask and grants the user the permissions that remain after filtering. Because an ACL does not have to contain an entry that applies to a user, if there is no match between the user requesting access to an object and any of the entries included in the object's ACL, the user is denied access to the object.

When the system evaluates an ACL, it evaluates the more-specific entries before it evaluates the less-specific entries; the order in which the entries appear in the ACL is irrelevant. Thus, the permissions granted to a group are applied to a user who is a member of the group only if the user's login name does not have its own entry. If an individual user granted permissions on an object is also granted additional permissions on the object by a group to which he or she belongs, the additional permissions are not recognized because the system evaluates the entry for the individual user first.

For example, suppose the user **frost** is specifically granted only the read and write permissions on a file. Suppose further that **frost** is also a member of the group **writers**, which is granted the read, write, and control permissions on the same file.

```
user:frost:rw----
group:writers:rw-c--
```

The system grants only the read and write permissions to **frost**; it does not grant the additional control permission to the user, because it evaluates the entry for the individual user before it evaluates the entry for the group. The system stops evaluating **frost**'s permissions before it checks the entry for the group to which **frost** belongs (the entry with more permissions).

You can use the ACL evaluation routine to extend or restrict the permissions granted to a particular member of a group. For instance, in the following example, the **purchasing** group is granted the read, write, and execute permissions on the file named **purchasing.memo**. If the user **terry** is a member of that group, you can grant **terry** more or fewer permissions on the file by including an entry for the user in the ACL for the file.

```
group:purchasing:rwx---
user:terry:r-----
```

The system grants the **r**, **w**, and **x** permissions to every member of the **purchasing** group except **terry**. Because it evaluates the more-specific entry (the one for the individual user **terry**) before it evaluates the less-specific entry (the one for the group **purchasing**), the system limits **terry** to just the **r** permission.

7.3.1 Interaction of ACLs with UNIX Mode Bits

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

Each class of user (**owner**, **group**, and **other**) can be assigned read (**r**), write (**w**), and execute (**x**) permissions or any combination of the three permissions via the appropriate mode bits. The operations associated with the bits are similar to those associated with the corresponding permissions for DCE ACLs. The settings of the mode bits for any object can be listed with the UNIX **ls -l** command or its equivalent; they can be set with the UNIX **chmod** command or its equivalent. Because DCE ACLs work only with DCE LFS data, mode bits are the only form of protection associated with non-LFS data.

When evaluating the ACL entries associated with a DCE LFS object, the system first reads the **owner**, **group**, and **other** mode bits for the object. It then uses these mode bits as the corresponding permissions in the appropriate ACL entries, as follows:

- The **owner** mode bits supply the **r**, **w**, and **x** permissions for the **user_obj** entry.

- The **other** mode bits supply the **r**, **w**, and **x** permissions for the **other_obj** entry.
- The **group** mode bits supply the **r**, **w**, and **x** permissions for the **mask_obj** entry. If the **mask_obj** entry does not exist, the **group** mode bits supply the three permissions for the **group_obj** entry. If the mode bits supply the permissions for the **mask_obj** entry, they do not affect the **group_obj** entry, and vice versa.

In effect, the UNIX mode bits override the ACL permissions for these entry types. They have no effect on the other ACL entry types.

For example, suppose a file owned by **terry** has the following ACL entries:

```
mask_obj:r-----
user_obj:rwxc--
group_obj:r-x---
other_obj:------
```

In this case, the corresponding UNIX mode bits for the file are

```
-rwxr----- 1 terry     3625 Nov 13 18:17 filename
```

The initial **r**, **w**, and **x** mode bits correspond to the matching permissions of the **user_obj** entry. The second **r** mode bit corresponds to the **r** permission of the **mask_obj** entry.

To keep an object's mode bits as closely aligned with its DCE ACLs as possible, the system modifies the appropriate mode bits associated with an object whenever you modify an **_obj** entry (**user_obj**, **mask_obj** or **group_obj**, or **other_obj**) in the object's ACL. Similarly, it effectively updates an object's ACL when you use the UNIX **chmod** command to modify the mode bits associated with the object. Thus, an object's mode bits are always kept in agreement with its ACL permissions.

7.4 Manipulating ACLs

The **acl_edit** command provided by the DCE Security Service is used to modify and list the permissions associated with ACL entries for file and directory objects. You can modify the permissions associated with any ACL you control (any ACL for which the control permission is associated with an entry that applies to you). Because the **user_obj** entry always has the control permission, you always control an object that you own (an object for which the **user_obj** entry applies to you).

To determine if you have the control permission for a file or directory object that you do not own, enter the **acl_edit -l** (list) command to list the object's ACL entries. (Note that, while having the control permission lets you modify the ACLs for an object, you must have the execute permission for a directory object to list the ACLs of the objects it contains.) Follow the ACL evaluation steps described in Section 7.3 to determine if you have the control permission. As described in that section, find the first entry from the following list that both applies to you and exists in the object's ACL:

- A **user** or **foreign_user** entry associated with your login name
- The **group_obj** entry, if you are a member of the group that owns the object, and all **group** and **foreign_group** entries for groups to which you belong
- The **other_obj** entry, if you are from the local cell
- A **foreign_other** entry associated with the cell to which you belong, if you are from a foreign cell
- The **any_other** entry

If the first entry (or entries in the case of multiple groups) that applies to you has the **c** permission (and the permission is not restricted by the **mask_obj** entry), you can use the **acl_edit** command with the **-m** (modify) subcommand to change the object's ACLs.

If an ACL entry already grants certain permissions to a user or group, permissions you grant with the **acl_edit** command *replace* the existing permissions; they are not added to the existing permissions. Therefore, if you want a user or group to retain the permissions already granted, you must include those permissions when you define the new ACL entry for the user or group. See Part 1B of this book for complete details about setting and using ACLs.

Chapter 8

Sharing Data in DFS

This chapter provides an overview of the methods available to share data in DFS. You can share both DCE LFS and non-LFS data by exporting your local disk to the DCE namespace. Exporting makes the data on the disk available for use by other DFS users in the DCE namespace, making your machine a Private File Server machine. You can also use DCE LFS data locally (on the local machine), regardless of whether it is exported to the DCE namespace. Because using exported DCE LFS data locally allows you to access the data via the local operating system of a machine, it increases the availability of the data in the event that the network becomes unavailable.

After DCE LFS data is exported, you can use ACLs to limit access to the data. While it is possible to export non-LFS data, most non-LFS file systems use only mode bits, not ACLs, for data protection. Refer to Chapter 7 for more information about using ACLs to protect exported DCE LFS data.

This chapter is intended only for users who administer their own workstations and want to share the data from the local disk of the machine with other users or want to locally access DCE LFS data exported from the local machine. It provides only an overview of the steps involved in these procedures. Complete details about exporting data and using exported data locally are included in the *OSF DCE Administration Guide*.

8.1 Exporting Data from Local Disks

You can share data stored on the local disk of a machine by exporting the DCE LFS aggregates and non-LFS partitions stored on the disk. Exporting an aggregate or partition makes the data it contains available to other users in the DCE namespace. Both DCE LFS aggregates and standard, non-LFS disk partitions can be exported. Exporting data from the local disk of a machine makes that machine a Private File Server machine.

Because a DCE LFS aggregate can store multiple filesets, exporting a DCE LFS aggregate lets you export multiple filesets. Conversely, because a non-LFS disk partition can store only a single fileset, exporting a non-LFS partition allows you to export only a single fileset.

Before exporting the filesets on an aggregate or partition, you must prepare the disk partition to be exported. For a non-LFS partition, this involves creating and mounting the partition locally. For a DCE LFS aggregate, this consists of using the DFS **newaggr** command to initially prepare the partition that is to house the aggregate. The **newaggr** command creates the metadata structure used by DCE LFS for ACL support, logging, and multiple fileset operations. It also creates temporary space used by the DCE LFS log for faster recovery from system failure. (The **newaggr** command is used to initialize an aggregate only once.)

Both DCE LFS and non-LFS data is exported by using the **dfsexport** command. This command reads the *dcelocal***/var/dfs/dfstab** file on the local disk of the machine to determine which partitions and aggregates can be exported. A partition or aggregate must have an entry in the **dfstab** file on the local machine if it is to be exported. Each entry provides information about a single partition or aggregate, such as its device name and its file system type.

When the **dfsexport** command reads the entry for a partition or aggregate from the **dfstab** file, it copies the entry to the *dcelocal***/var/dfs/dfsatab** file, which it creates on the local disk of the machine. The **dfsatab** file lists the devices already exported to the DCE namespace. The **dfsexport** command will not export a partition or aggregate that has an entry in the **dfsatab** file. The entry for a partition or aggregate must be removed from the **dfsatab** file before the device can be exported subsequent times (for example, after the machine is rebooted).

The **dfsexport** command is generally included in a workstation's initialization file (**/etc/rc** or its equivalent) to automatically export all desired partitions and aggregates when the machine is restarted. The initialization file also typically includes a command such as **rm** (or its equivalent) to remove the **dfsatab** file before it executes the **dfsexport** command.

Finally, for data on an exported partition or aggregate to be available in the DCE namespace, the fileset on which it resides must have an entry in the Fileset Location Database (FLDB), and the fileset must have a mount point in the DCE namespace. The following subsections list the prerequisites to exporting data from the disk of a local machine. They also provide detailed information about exporting both DCE LFS and non-LFS data to the DCE namespace.

Note: The information presented in these subsections is intended only for users who administer their own machines. A user must have specific administrative privileges to perform these operations.

8.1.1 Prerequisites to Exporting Data

A number of prerequisites must be met before you can export either a DCE LFS aggregate or a non-LFS disk partition from a machine. The following lists describe the preliminary conditions that must be true before a machine can export data to the global namespace.

The following processes must be running in the cell:

- A CDS server process (**cdsd**)
- A Security Server process (**secd**)
- A Fileset Location Server (FL Server) process (**flserver**)

The following conditions relative to the machine from which data is to be exported must also be met:

- A CDS advertiser process (**cdsadv**) must be running on the machine.
- A CDS clerk process (**cdsclerk**) must be running on the machine. (The **cdsadv** process automatically spawns a new **cdsclerk** process for each user on the machine.)

- A Security Client process (**sec_clientd**) must be running on the machine.
- An RPC process (**rpcd**) must be running on the machine.
- An RPC binding must exist for the DCE pathname of the machine in CDS.
- A DFS server principal must exist for the machine in the registry database.
- A server entry must exist for the machine in the FLDB.
- A key file and a key must exist on the machine.
- The **dfsbind** process must be running on the machine.
- The File Exporter must be running in the kernel of the machine. (The File Exporter is started with the **fxd** process.)
- The Fileset Server process (**ftserver**) must be running on the machine.
- The Replication Server process (**repserver**) must be running on the machine.
- The **upclient** process (the client portion of the Update Server) must be running on the machine to retrieve binary files from the proper Binary Distribution machine.
- The Basic OverSeer Server process (**bosserver**) must be running on the machine.

To be able to administer the filesets on the machine, you must also have the following administrative privileges on the machine:

- You must be included in the **admin.ft** administrative list on the machine. Inclusion in this list allows you to manipulate filesets exported from the machine.
- You must be included in the group that has ownership of the server entry for the machine in the FLDB. Inclusion in this group allows you to modify the FLDB entries for filesets exported from the machine.
- You must be included in the administrative group for the File Exporter on the machine. Inclusion in this group enables you to modify the permissions of all data exported from the machine.

Once all of these prerequisites are satisfied, the steps described in the following subsections can be used to export DCE LFS aggregates and non-LFS partitions from the machine. The previous lists are intended primarily as a checklist to the prerequisites for exporting data. Detailed information on satisfying the prerequisites appears in the *OSF DCE Administration Guide*.

The commands referenced in the following sections, like those used to meet the prerequisites listed previously, are administrative-level commands. You must have the necessary administrative privileges (inclusion in the administrative lists mentioned previously) and ACL permissions to execute them.

8.1.1.1 Exporting DCE LFS Data

Execute the following steps to export a DCE LFS aggregate to the DCE namespace. Once the aggregate is exported, you can use the **fts create** command to create and register filesets on the aggregate, after which you can use the **fts crmount** command to mount the new filesets. You can also use the **fts setquota** command to change the default quotas of the filesets and the **acl_edit** command to modify the default ACLs of the root directories of the filesets.

The following instructions assume the aggregate was already initialized with the **newaggr** command sometime prior to exporting. The **newaggr** command formats a partition for use as a DCE LFS aggregate. Do not issue the **newaggr** command on a partition that contains data you want to retain. The command destroys all data on the specified partition. Also, do not issue the **newaggr** command to create non-LFS aggregates.

Caution: Do not issue the **newaggr** command on a locally mounted partition. Using **newaggr** on a locally mounted partition causes the kernel of the machine to panic.

1. Verify that you have the write, execute, and insert permissions for the directories in which mount points for any exported filesets are to be created. If necessary, issue the **acl_edit -l** command to check the ACL permissions for the directories.

2. Log in as **root** on the machine from which the aggregate is to be exported.

3. Use a text editor to edit the **dfstab** file on the local disk of the machine to include the required information for the aggregate to be

exported. For each aggregate to be exported, the **dfstab** file must contain a single line containing the following fields, in the order listed. Each field must be separated by a minimum of one space or tab.

- The block device name of the aggregate; for example, **/dev/lv03**.
- The aggregate name to be associated with the exported aggregate. An aggregate name can contain any characters, but it can be no longer than 31 characters, and it must be unique within the **dfstab** file. Choose a short, descriptive name; for example, **lfs1**.
- The **lfs** identifier to indicate that a DCE LFS aggregate is to be exported.
- A positive integer, different from any other aggregate ID in the **dfstab** file, to act as the aggregate ID for the aggregate; for example, **2** or **7**.

The following example shows a typical entry for a DCE LFS aggregate in a **dfstab** file:

```
/dev/lv03  lfs1  lfs  2
```

4. Issue the **dfsexport** command, as follows, to export the aggregate. This command, which is usually included with its **-all** option in a machine's initialization file (**/etc/rc** or its equivalent), reads the **dfstab** file to determine the aggregates and partitions available to be exported. It then exports the devices specified with its options. Omit all of the command's options to list the aggregates and partitions with entries in the **dfsatab** file.

   # dfsexport [{**-all** | **-aggregate** *name*}] [**-type** *name*]

   The **-all** option specifies that all aggregates and partitions listed in the **dfstab** file are to be exported. Use this option with **-type** to export only DCE LFS aggregates or only non-LFS partitions. Use this option or use **-aggregate**.

   The **-aggregate** *name* option specifies the device name or aggregate name of a specific aggregate or partition. These names are specified in the first and second fields of the entry for the aggregate or partition in the **dfstab** file. Use this option or use **-all**.

The **-type** *name* option specifies the file system type to be exported. Specify **lfs** to export only DCE LFS aggregates; specify **ufs** to export only non-LFS partitions. Use this option only with **-all**; omit it and use **-all** to export all aggregates and partitions.

5. Log out as **root** from the machine to return to your authenticated DCE identity.

6. Issue the **fts create** command, as shown below, to create a fileset on the aggregate and to register the new fileset in the FLDB. The FL Server allocates a unique fileset ID number for the fileset.

   % **fts create -ftname** *name* **-server** *machine* **-aggregate** *name*

   Use the **-ftname** *name* option to specify the complete name you want to assign to the fileset to be created. The name must be unique in the FLDB of the cell in which the fileset is to be created. It can be no longer than 102 characters, and it can include alphanumeric characters, **.** (periods), **-** (dashes), and _ (underscores). It must include at least one alphabetic character or an _ (underscore).

   The **-server** *machine* option specifies the name of the machine on which the exported aggregate resides. Specify the name of the machine as a DCE pathname (for example, **/.../abc.com/hosts/fs1**).

   The **-aggregate** *name* option gives the device name, aggregate name, or aggregate ID of the exported aggregate on which the fileset is to be created. These names appear in the first, second, and fourth fields of the aggregate's entry in the **dfstab** file.

   Repeat the **fts create** command for each fileset you want to create on the aggregate.

7. Enter the **fts crmount** command, as shown below, to create a mount point in the file system for the new fileset. This makes the contents of the fileset visible to other users. After the fileset is mounted, its pathname can use the global **/...** designation.

   % **fts crmount -dir** *directory_name* **-fileset** {*name* | *ID*}

   The **-dir** *directory_name* option gives the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the current working directory.

The **-fileset** option specifies the complete name or fileset ID number of the fileset to be mounted.

Repeat the **fts crmount** command for each fileset created in Step 6.

Once these steps are performed, filesets on the exported aggregate are available to all authorized users in the DCE namespace. The following two steps are optional. They can be used to change the default quota or ACL permissions of any DCE LFS fileset.

8. If necessary, enter the **fts setquota** command to change the default quota (5000 kilobytes) of the new fileset:

% **fts setquota** {**-path** {*filename* | *directory_name*} | **-fileset** {*name* | *ID*}} **-size** *kbytes*

   The **-path** option specifies the name of a file or directory on the fileset whose quota you want to set. Use this option or use **-fileset**.

   The **-fileset** option specifies the complete name or fileset ID number of the fileset whose quota you want to set. Use this option or use **-path**.

   The **-size** option specifies the maximum amount of disk space (*kbytes*) the fileset can occupy. Specify the value in kilobytes.

   Repeat the **fts setquota** command for each fileset whose quota you want to change.

9. Use the **acl_edit** command with its **modify** (**-m**) subcommand to change the default ACLs of the root directory of any fileset. See Part 1B of this book for more information about setting and examining ACLs.

8.1.1.2 Exporting Non-LFS Data

Perform the following steps to export a non-LFS disk partition to the DCE namespace. The following instructions assume the partition is mounted locally and that its name appears in the **fstab** file (or its equivalent).

1. Verify that you have the write, execute, and insert permissions for the directory in which you want to create the mount point for the fileset. If necessary, issue the **acl_edit -l** command to check the ACL

permissions for the directory. (You need to have the write and execute permissions if the directory is in a non-LFS fileset.)

2. Use the **fts crfldbentry** command, as shown below, to register the single non-LFS fileset on the partition in the FLDB. Issuing this command allows the FL Server to track the location of the fileset in the same manner that it tracks the locations of DCE LFS filesets.

   The command returns the fileset ID numbers allocated by the FL Server for the read/write, read-only, and backup versions of the fileset. Because a non-LFS fileset cannot have read-only and backup versions, ignore the latter two numbers. Use the read/write ID number returned by the command as the fileset ID in the **dfstab** file.

   % fts crfldbentry -ftname *name* **-server** *machine* **-aggrid** *ID*

   The **-ftname** *name* option is the complete name you want to assign to the fileset on the partition to be exported. The name must be unique in the FLDB of the cell in which the fileset is located. It can be no longer than 102 characters, and it can include alphanumeric characters, **.** (periods), **-** (dashes), and _ (underscores). It must include at least one alphabetic character or an _ (underscore).

   The **-server** *machine* option specifies the name of the machine on which the partition resides. Specify the name of the machine as a DCE pathname (for example, **/.../abc.com/hosts/fs1**).

   The **-aggrid** *ID* option is a positive integer to serve as the aggregate ID for the partition to be exported. The integer must be different from any other aggregate ID in the **dfstab** file on the machine on which the partition resides.

3. Log in as **root** on the machine from which the partition is to be exported.

4. Use a text editor to edit the **dfstab** file on the local disk of the machine to include the necessary information for the partition to be exported. For each partition to be exported, the **dfstab** file must contain a single line containing the following fields, in the order listed. Each field must be separated by a minimum of one space or tab. Note that, because an exported non-LFS partition can contain only a single fileset, you include a fileset ID number with the partition's entry in the **dfstab** file.

- The block device name of the partition; for example, **/dev/lv02**.
- The aggregate name to be associated with the exported partition. The aggregate name of a non-LFS partition must match the name of its local mount point (for example, **/usr**). An aggregate name can contain any characters, but it can be no longer than 31 characters, and it must be unique within the **dfstab** file. Choose a short, descriptive name.
- The **ufs** identifier to indicate that a non-LFS file system is to be exported.
- A positive integer, different from any other aggregate ID in the **dfstab** file, to serve as the aggregate ID for the partition; for example, **3** or **10**. This integer must match the one specified with the **-aggrid** option of the **fts crfldbentry** command in the previous step.
- The unique fileset ID number returned by the **fts crfldbentry** command in the previous step; for example, **0,,12262**. Use the read/write ID number (not the read-only or backup ID number) returned by the command as the value for this field.

The following example displays a typical entry for a non-LFS partition in a **dfstab** file:

```
/dev/lv02  /usr  ufs  3  0,,12262
```

5. Issue the **dfsexport** command, as shown below, to export the partition. This command reads the **dfstab** file to determine the partitions and aggregates available to be exported. It then exports the devices specified with its options. (The **dfsexport** command is typically included with its **-all** option in a machine's initialization file — **/etc/rc** or its equivalent.) Omit all of the command's options to list the aggregates and partitions with entries in the **dfsatab** file.

   % **dfsexport** [{**-all** | **-aggregate** *name*}] [**-type** *name*]

   The **-all** option specifies that all aggregates and partitions listed in the **dfstab** file are to be exported. Use this option with **-type** to export only non-LFS partitions or only DCE LFS aggregates. Use this option or use **-aggregate**.

The **-aggregate** *name* option specifies the device name or aggregate name of a specific partition or aggregate. These names are specified in the first and second fields of the entry for the partition or aggregate in the **dfstab** file. Use this option or use **-all**.

The **-type** *name* option gives the file system type to be exported. Specify **ufs** to export only non-LFS partitions; specify **lfs** to export only DCE LFS aggregates. Use this option only with **-all**; omit it and use **-all** to export all aggregates and partitions.

6. Log out as **root** from the machine to return to your authenticated DCE identity.

7. Enter the **fts crmount** command, as shown below, to create a mount point in the file system for the new fileset. This makes the contents of the fileset visible to other users. After the fileset is mounted, its pathname can use the global **/...** designation.

   % **fts crmount -dir** *directory_name* **-fileset** {*name* | *ID*}

   The **-dir** *directory_name* option gives the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace. Include a complete pathname unless you want to mount the fileset in the current working directory.

   The **-fileset** option specifies the complete name or fileset ID number of the fileset to be mounted.

When these steps are complete, the fileset on the exported partition is available to all authorized users in the DCE namespace.

8.2 Using DCE LFS Data Locally

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

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

Mounting a DCE LFS fileset locally does not improve local access time to the data on the fileset if the fileset is accessed via the namespace. However, access time to the data on the fileset is improved if the fileset is accessed via the local operating system.

Availability of the fileset is generally improved because the fileset can still be accessed locally in the event of a network outage. If data on a DCE LFS fileset physically located on the local disk of your machine is available only through the namespace, a network outage makes it impossible to access the data. Mounting a DCE LFS fileset locally allows you to continue to work with the data on the fileset in the event of a network failure because you can still access the data via the operating system on your local machine.

Provided the DCE LFS aggregate containing the fileset is exported and the fileset is mounted in the DCE namespace, you can also access a locally mounted DCE LFS fileset globally. The pathname associated with the fileset is different for local and global access. For example, a fileset accessed in the local operating system as **/lfs/jlw** may be accessed in the DCE namespace as **/.../abc.com/fs/usr/jlw**.

The following subsection describes the steps involved in mounting a DCE LFS fileset locally. It is assumed that the aggregate that houses the fileset has already been initialized with the **newaggr** command and that the fileset to be made available locally has already been created with the **fts create** command.

8.2.1 Making DCE LFS Data Available Locally

To mount a DCE LFS fileset locally, perform the following steps:

1. Log in as **root** on your workstation. For example:

   ```
   % su root
   Password: root_password
   ```

2. Create an empty directory in which the fileset is to be locally mounted. For example:

   ```
   # mkdir directory_name
   ```

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

   # mount *device_name directory_name fileset_ID*

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

Part 2

DCE User's Reference

Chapter 9

DCE Security Service

intro

Purpose Introduction to the DCE Security user commands

Description

This section describes publicly accessible DCE Security commands. These commands are as follows:

- The **acl_edit** command, which manages Access Control Lists (ACLs) for DCE objects.
- The **dce_login** command, which validates a principal's identity and obtains a principal's network credentials. This command is used primarily during DCE configuration. Use the login utility supplied by your platform vendor for user login.
- The **kinit** command, which obtains and caches a ticket granting ticket.
- The **klist** command, which lists cached tickets.
- The **kdestroy** command, which destroys your login context and credentials.
- The **chpass** command, which changes user information, such as login name, password, home directory, password and account expiration dates, and login shell.
- The **su** command, which allows you to assume another user's identity.

See the command's reference page for further information on each command.

acl_edit

Purpose

Edits or lists an object's ACLs

Synopsis

acl_edit {[**-e**] *pathname* | **-addr** *string_binding component_name*} [**-ic** | **-io**] [**-n** | **-c**] [*command_line_subcommands*] [**-ngui**] [**-v**]

Options

-e *pathname* Specifies that the ACL on the Directory Service entry is to be edited. You must specify the *pathname* argument if you use the **-e** option.

The **-e** option is especially useful in case of an ambiguous *pathname*. The *pathname* argument can be interpreted in two ways if it is the name of a leaf object in the Directory Service (that is, if it is not the name of a directory). It can be interpreted as the Directory Service entry itself, or as the object (whatever it is) referenced by that Directory Service entry. When such a *pathname* is specified, the **-e** option directs **acl_edit** to the ACL on the Directory Service entry.

-addr *string_binding component_name*
The **-addr** option lets you identify the object whose ACLs you want to edit by supplying the RPC binding handle of the ACL Manager that controls access to the object (with the *string_binding* argument) and the relative pathname of the object (with the *component_name* argument). Because you have identified the RPC binding handle, you can specify only the object's relative pathname for *component_name*.

The most common way to identify the object whose ACLs you want to manipulate is through the *pathname* argument, described below. The **-addr** option is used primarily by applications that do not use the Directory Service, but do use the generic ACL Manager. It can also be used if the Directory Service is unavailable.

-ic For container objects only, specifies that the object's Initial Container Creation ACL is to be edited. The Initial Container Creation ACL is applied by default to any containers created within the ACL'd container. If this option is specified and the object named in *pathname* is not a container, an error is returned.

-io For container objects only, specifies that the object's Initial Object Creation ACL is to be edited. The Initial Object Creation ACL is applied by default to any simple objects (that is, objects that are not containers) created within the ACL'd container. If this option is specified and the object is not a container, an error is returned.

-n Specifies that a new mask should not be calculated. This option is useful only for objects that support the **mask_obj** entry type and that recalculate a new mask after they are modified.

-c Creates or modifies the object's **mask_obj** type entry with permissions equal to the union of all entries other than type **user_obj**, **other_obj**, and **unauthenticated**. This creation or modification is done after all other modifications to the ACL are performed. The new mask is set even if it grants permissions previously masked out. It is recommended that you use this option only if not specifying it results in an error. This option is useful only for objects that support the **mask_obj** entry type and that recalculate a new mask after they are modified.

If you specify the **-c** option for an ACL that does not support **mask_obj** entry type, **acl_edit** returns an error when it attempts to save the ACL, aborting all subcommands supplied on the command line.

-ngui Specifies that a Graphical User Interface (GUI) should not be used even if a GUI is available. By default, when **acl_edit** is invoked in interactive mode, it invokes a GUI if one is available and if the terminal is capable of using the GUI. If a GUI is not available, or the terminal is not capable of using the GUI, **acl_edit** invokes the interactive interface.

-v Run in verbose mode.

Arguments

pathname The full pathname of the object whose ACL is to be viewed or edited. If the object is in another cell, *pathname* must be fully qualified to include the cell identifier.

command_line_subcommands
The command-line subcommands, which act on the object specified by *pathname*, are entered as part of the command string that invokes **acl_edit**. Only one command-line subcommand can be specified per invocation. The commands follow. See the description of the equivalent interactive subcommand for a more detailed description of the command functions.

-m *acl_entries*
Adds a new ACL entry or changes the permissions of an existing entry. You can enter multiple entries, each separated by a space.

-p
Purges all masked permissions (before any other modifications are made). This option is useful only for ACLs that contain an entry of type **mask_obj**. Use it to prevent unintentionally granting permissions to an existing entry when a new mask is calculated as a result of adding or modifying an ACL entry.

-d *acl_entries*
Deletes an existing entry from the ACL associated with the specified object. You can enter multiple entries, each separated by a space.

-s *acl_entries*
Replaces (substitutes) the ACL information associated with this object with *acl_entries*. All existing entries are removed and replaced by the newly specified entries. If you specify the **-s** subcommand, you cannot specify the **-f** or **-k** subcommand. You can enter multiple entries, each separated by a space.

-f *file*
Assigns the ACL information contained in *file* to the object. All existing entries are removed and replaced by the entries in the file. If you specify the **-f** subcommand, you cannot specify the **-s** or **-k** subcommand.

-k
Removes all entries, except entries of type **user_obj** (if they are present). If you specify the **-k** subcommand, you cannot specify the **-f** or **-s** subcommand.

-l
Lists the entries in the object's ACL.

The command-line subcommands are evaluated in the following order:

1. **-p**
2. **-s** or **-f** or **-k**
3. **-d**

4. **-m**
5. **-l**

Description

The **acl_edit** command is a client program that, when invoked, binds to the specified object's ACL Manager (which is implemented in the object's server), and allows the user to manipulate the object's ACL through the standard DCE ACL interface. This interface is the **sec_acl_...()** interface documented in the *OSF DCE Application Development Reference*.

The **acl_edit** command automatically binds to the server of the object specified, and then communicates (through the standard DCE ACL interface) with that server's ACL manager in response to user input.

Exactly what the ''object specified'' is depends partly on whether or not the **-e** option is specified. Specifying **-e** means that you want to operate on the Directory Service ACL. In other words, you want **acl_edit** to bind to the CDS server and allow you to operate on the ACL maintained by that server on the object's directory entry. If, on the other hand, you want to access the ACL on the object itself — that is, the ACL on the object to which the directory entry refers — then you simply omit the **-e** option. The result will be that **acl_edit** will bind to that object's server (the server must, of course, implement an ACL manager), giving you access to the object's ACL.

All **acl_edit** subcommands act on the object specified by *pathname* when you invoked **acl_edit**. You can invoke **acl_edit** in either command-line or interactive mode:

- To invoke **acl_edit** in command-line mode, enter the command, the object's pathname, options, and the command-line subcommand on the line that invokes **acl_edit**. Only one command-line subcommand can be entered per **acl_edit** invocation.

- To invoke **acl_edit** in interactive mode, enter only **acl_edit**, the object's pathname, and options. The **acl_edit** prompt is then displayed. In this mode, you enter interactive subcommands that let you edit and view entries in the object's ACL and view help information about the **acl_edit** command itself.

Interactive Subcommands

The following subcommands are available when **acl_edit** is invoked in interactive mode. All of the commands act on the ACL associated with the object specified by *pathname* when **acl_edit** was invoked.

?	Displays the available **acl_edit** subcommands.
ab[ort]	Exits **acl_edit** without saving the changes to the object's ACL.
as[sign] *filename*	Applies the ACL entries in *filename* to the specified object. This subcommand removes existing entries and replaces them with the entries in the file.
c[ell] *name*	Sets the cell name to be associated with the ACL. This subcommand is used primarily to facilitate copying ACLs to different cells. The default cell name stays in place until you run the subcommand again to change it.
co[mmit]	Saves all changes to the ACL without exiting.
d[elete] *acl_entry*	Deletes the specified ACL entry.
e[xit]	Exits from **acl_edit**, saving any changes to the object's ACL.
g[et_access]	Displays the permissions granted in the specified object's ACL to the principal that invoked **acl_edit**.
h[elp] [*command* ...]	Initiates the **help** facility. If you enter only the command **help**, **acl_edit** displays a list of all commands and their functions. If you enter **help** and a command (or commands separated by a space), **acl_edit** displays help information on the specified commands. Entering **help sec_acl_entry** displays information about ACL entries.
k[ill_entries]	Removes all ACL entries except the **user_obj** entry if it exists.
l[ist]	Lists the entries in the object's ACL.
m[odify] *acl_entry* [**-n** | **-c**]	Adds a new ACL entry or replaces an existing ACL entry. This command affects a single ACL entry. To add or replace all of an object's ACL entries, see the **su[bstitute]** subcommand. For objects that calculate a new mask when their ACLs are modified, the **-n** option specifies that a new mask should not be calculated; the **-c** option specifies that the object's **mask_obj** entry should have permissions equal to the union of all entries other than **user_obj**, **other_obj**, and **unauthenticated**. The mask is calculated after the ACL is modified.

If you use the **-c** option, the new mask is set even if it grants permissions previously masked out. It is recommended that you use the **-c** option only if not specifying it results in an error. If the new mask unintentionally grants permissions to an existing entry, the modify operation causing the mask recalculation will abort with an error unless you specify either the **-c** or **-n** option.

p[ermissions]
Lists the available permission tokens and explanations.

pu[rge] Purges all masked permissions. This option is useful only for ACLs that contain an entry of type **mask_obj**. Use it to prevent unintentionally granting permissions to an existing entry when a new mask is calculated as a result of adding or modifying an ACL entry.

su[bstitute] *acl_entry* [*acl_entry* ...]
Replaces all ACL entries with the one or ones specified. This subcommand removes all existing entries and adds the ones specified by *acl_entry*. To replace only a single ACL entry, see the **m[odify]** subcommand.

t[est_access] [*permissions* ...]
Tests whether or not the permissions specified in the command are granted to the principal under whose DCE identity the **acl_edit** command was invoked. The option returns `Granted` if the permissions are granted or `Denied` if they are not.

ACL Entries

An ACL entry has the following syntax:

type[:*key*]:*permissions*

where:

type	Identifies the role of the ACL entry.
key	Identifies the specific principal or group to whom the entry applies. For an entry type of **extended**, *key* contains the ACL data.
permissions	The ACL permissions.

A thorough description of each syntax component follows.

Type

The *type* tag identifies the role of the ACL entry. Valid types are the following:

- **user_obj** - Permissions for the object's real or effective user.
- **group_obj** - Permissions for the object's real or effective group.
- **other_obj** - Permissions for others in the local cell who are not otherwise named by a more specific entry type.
- **user** - Permissions for a specific principal user in the ACL's cell. This type of ACL entry must include a key that identifies the specific principal.
- **group** - Permissions for a specific group in the ACL's cell. This type of ACL entry must include a key that identifies the specific group.
- **foreign_user** - Permissions for a specific, authenticated user in a foreign cell. This type of ACL entry must include a key that identifies the specific principal and the principal's cell.
- **foreign_group** - Permissions for a specific, authenticated group in a foreign cell. This type of ACL entry must include a key that identifies the specific group and the group's cell.
- **foreign_other** - Permissions for all authenticated principals in a specific foreign cell, unless those principals are specifically named in an ACL entry of type **foreign_user** or members in a group named in an entry of type **foreign_group**. This type of ACL entry must include a key that identifies the specific foreign cell.
- **any_other** - Permissions for all authenticated principals unless those principals match a more specific entry in the ACL.
- **mask_obj** - Permissions for the object mask that is applied to all entry types except **user_obj**, **other_obj**, and **unauthenticated**.
- **unauthenticated** - Maximum permissions applied when the accessor does not pass authentication procedures. This entry is used for principals that have failed authentication due to bad keys, principals who are entirely outside of any authentication cell, and principals who choose not to use authenticated access. Permissions granted to an unauthenticated principal are masked with this entry, if it exists. If this entry does not exist, access to unauthenticated principals is always denied.
- **extended** - A special entry that allows client applications running at earlier DCE versions to copy ACLs to and from ACL Managers running at the current DCE version without losing any data. The **extended** entry allows

the application running at the lower version to obtain a printable form of the ACL. The **extended** ACL entry has the following form:

extended: *uuid.ndr.ndr.ndr.ndr.number_of_byte.data*

where:

uuid
: Identifies the type extended ACL entry. (This UUID can identify one of the ACL entry types described here or an as-yet-undefined ACL entry type.)

ndr.ndr.ndr.ndr
: Up to three Network Data Representation (NDR) format labels (in hexadecimal format and separated by periods) that identify the encoding of data.

number_of_bytes
: A decimal number that specifies the total number of bytes in *data.*

data
: The ACL data in hexadecimal form. (Each byte of ACL data is two hexadecimal digits.) The ACL data includes all of the ACL entry specifications except the permissions (described later) that are entered separately. The data is not interpreted; it is assumed that the ACL Manager to which the data is being passed can understand that data.

Key

The *key* identifier (principal or group name) specifies the principal or group to which the ACL entry applies. For entries of entry type **extended,** *key* is the data passed from one ACL Manager to another. A *key* is required for the following types of ACL entries:

- **user** - Requires a principal name only.
- **group** - Requires a group name only.
- **foreign_user** - Requires a fully qualified cell name in addition to the principal name.
- **foreign_group** - Requires a fully qualified cell name in addition to the group name.
- **foreign_other** - Requires a fully qualified cell name.

Permissions

The *permissions* argument specifies the set of permissions that defines the access rights conferred by the entry. Since each ACL Manager defines the permission tokens and meanings appropriate for the objects it controls, the actual tokens and their meanings vary.

For example, the Distributed File Service, the Directory Service, and the Security Registry Service each implement a separate ACL Manager, and each can use a different set of tokens and permissions. This means that file system objects, objects in the namespace, and registry objects could each use different permissions. Use the **p[ermissions]** subcommand to display the currently available tokens and their meanings. See the documentation for the DCE component you are using to obtain a more detailed description of its specific permissions.

Examples

1. The following example uses the interactive interface to set permissions for the **unauthenticated** and **mask_obj** entry type:

```
sec_acl_edit> m mask_obj:rwx
sec_acl_edit> m unauthenticated:r
```

2. The following example uses the interactive interface to set permissions for the effective user, group, and others in the ACL's cell:

```
sec_acl_edit> m user_obj:crwx
sec_acl_edit> m group_obj:rwx
sec_acl_edit> m other_obj:rwx
```

3. The following example uses the command-line interface to invoke **acl_edit** and assign permissions for the file **progress_chart** to the authenticated user **mike** in the local cell:

```
% acl_edit /.../dresden.com/afs/walden/progress_chart -m user:mike:crwx
```

   Note that because this entry will be filtered through the object mask (**mask_obj**), which specifies only **rwx** permissions, the actual permissions will be **rwx**, not **crwx**. The **l(ist)** subcommand will show those permissions as follows:

```
user:mike:crwx  #effective -rwx---
```

4. The following example uses the interactive interface to set permissions for the authenticated foreign user named **burati** in the cell named **/.../usc-cs.uscal.edu**:

```
sec_acl_edit> m foreign_user:/.../usc-cs.uscal.edu/sailing/staff/burati:rwx
```

5. The following example uses the non-interactive command-line interface to invoke **acl_edit** and set the Initial Container Creation permissions for the directory that is named **walden**:

```
% acl_edit /.../dresden.com/afs/walden -ic -m /user:walden:crwxid
```

chpass

Purpose

Changes user database information

Synopsis

chpass [**-a** *list* | **-s** *shell* | *user*] [**-l** | **-n**]

Options

-a *list*

Supplies user database information on the command line, instead of being prompted. The *list* argument is a colon-separated list of all user database fields in the format specified in the **passwd(5)** reference page. Although you must enter the : (colon) separators for each field, you can leave non-required fields empty.

The option is available only if you have the appropriate rights.

-s *shell*

Changes the user's login shell to the one specified in *shell.*

-l

When you change a password, command options let you specify whether the changes affect the network registry or only the local password override file. The **-l** option specifies that the change take place only on the local override file and not in the network registry. If overrides exist for you and you do not enter the **-l** or **-n** option, you will be prompted for the option.

-n

When you change a password, command options let you specify whether the changes affect the network registry or only the local password override file. The **-n** option specifies that the change take place only on the network registry and not in the local override file. If overrides exist for you and you do not enter the **-l** or **-n** option, you will be prompted to enter one of them.

Arguments

user

The *user* argument indicates the user whose database information you want to change. If omitted, the user is the current user.

Description

The **chpass** command changes user database information associated with *user* or, if the *user* argument is omitted, the current user.

Note that the functionality of the **chpass** command as described in this reference page can change depending on the platform on which you are running the command. Each platform vendor integrates this command (based on 4.4BSD source) with the vendor's own login facility.

You can edit information associated with *user* only if you are *user* or have the appropriate rights.

If you are logged in as the superuser, you can specify all changes on the command line in the format described in the **passwd(5)** reference page. Otherwise, except for changes to the login shell, which can be specified on the command line, **chpass** prompts for the information you are allowed to change. Depending on your rights, the information will include all or a subset of the following list:

- **Login** - The login name used to access the account. Because the login name or the UNIX ID controls file access, they must be unique within the cell. In multicell environments, this uniqueness is ensured by automatically appending the cell designator to the user's name.

  While it is possible to have multiple entries with identical login names or identical user IDs, it is usually a mistake to do so. Routines that manipulate these files will often return only one of the multiple entries, and that one by random selection.

- **Password** - The encrypted account password.

- **UNIX ID** - The UNIX ID associated with the login name.

- **Gid** - The group that the user will be placed in at login. Since this system supports multiple groups (see **groups(1)**), this field currently has little special meaning. This field may be filled in with either a number or a group name.

- **Change** - The date on which the user's password expires. Entered in the form *mmm dd yyyy* where *mmm* is the first 3 characters of the month name, *dd* is the day of the month, and *yyyy* is the year.

- **Expire** - The date on which the account expires. Entered in the form *mmm dd yyyy* where *mmm* is the first 3 characters of the month name, *dd* is the day of the month, and *yyyy* is the year.

- **Class** - The user's general classification. This **class** field is currently unused.

- **Home Directory** - The full UNIX pathname of the directory in which the user will be placed at login.

- **Shell** - The user's login shell. If the **shell** field is empty, the Bourne shell, **/bin/sh**, is assumed. Only the superuser can change the shell (whether it is standard or nonstandard) to a nonstandard shell. Nonstandard is defined as a shell not found in **/etc/shells**.
- **Full Name** - The user's full name.
- **Location** - The user's office location.
- **Home Phone** - The user's home phone number.
- **Office Phone** - The user's office phone number.

Once the information has been verified, the network registry is updated.

Environment Variables

The information displayed by **chpass** is formatted and supplied to an editor. When the editor terminates, the information is reread and used to update the user database itself. The **vi(1)** editor is used, unless the **EDITOR** environment variable is set to an alternative editor.

Files

/etc/master.passwd	The user database.
/etc/shells	The list of approved shells.

Related Information

Commands: **login(1)**, **finger(1)**, **getusershell(3)**, **passwd(5)**, **passwd.override(8sec)**.

dce_login

Purpose

Validates a principal's identity and obtains the principal's network credentials

Synopsis

dce_login [*principal_name*] [*password*] [**-c**]

Options

-c	Causes the principal's identity to be certified. If you do not specify **-c**, the principal's identity is validated only.

Arguments

principal_name	The name of the principal to log in as.
password	The password for *principal_name*.

Description

The **dce_login** command is supplied for use in DCE configuration. It validates a principal's identity and obtains the principal's network credentials. If the **-c** option is supplied, the command also certifies the principal's identity, and, if the principal is able to be certified, creates an entry for the principal in the machine's local registry. If the principal is not able to be certified, the command attempts to log the principal in via the local registry.

The command executes the shell specified in the **SHELL** environment variable.

The *principal_name* argument specifies the name of the principal who is logging in. The *password* argument specifies the principal's password. If you do not supply a principal name or a principal password, **dce_login** prompts for them. If you enter them both on the command line, you must specify the principal name first, followed by the password.

kdestroy

Purpose Destroys a principal's login context and associated credentials

Synopsis **kdestroy** [**-c** *cache_name*]

Options

-c *cache_name*	Specifies that the login context and associated credentials for *cache_name* should be destroyed instead of the default cache.

Description

The **kdestroy** command destroys a principal's login context and the principal's credentials. Until the credentials are reestablished by either executing the **dce_login** command or the **kinit** command, the principal and any processes created by the principal will be limited to unauthenticated access.

Files

/tmp/krb5cc_[*unix_id*]	If the **KRB5CCNAME** environment variable is set, the default credentials cache. ([*unix_id*] is the decimal UNIX ID of the user.)

Related Information

Commands: **klist(1sec)**, **kinit(1sec)**.

kinit

Purpose

Obtains and caches ticket-granting ticket

Synopsis

kinit [**-c** *cachename*] [**-f**] [**-l** *lifetime*] [**-p**] [**-r** *lifetime*] [**-v**] [*principal*]

Options

-c *cachename* Specifies an alternative credentials cache, *cachename*, to be used in place of the default credentials cache. The **kinit** command overwrites the contents of the alternative cache with the current credentials.

The name of the default credentials cache may vary between systems. However, if the **KRB5CCNAME** environment variable is set, its value is used to name the default cache.

-f Requests the FORWARDABLE option. This option allows a ticket-granting ticket with a different network address than the present ticket-granting ticket to be issued to the principal. For forwardable tickets to be granted, the principal's account in the registry must specify that the principal can be granted forwardable tickets.

-l *lifetime* Specifies the lifetime of the ticket-granting ticket in hours. If this option is not specified, the default ticket lifetime (set by each site using the **rgy_edit(1sec)** command) is used.

-p Requests the PROXIABLE option. This option allows a ticket with a different network address than the present ticket to be issued to the principal. For proxiable tickets to be granted, the principal's account in the registry must specify that the principal can be granted proxiable tickets.

-r *lifetime* Requests the RENEWABLE option. This option allows the tickets issued to the principal to be renewed. For renewable tickets to be granted, the principal's account in the registry must specify that the principal can be granted renewable tickets. The lifetime of the ticket-granting ticket is specified in hours by *lifetime*.

-v Specifies that the command should run in verbose mode.

Arguments

principal The *principal* argument specifies the name of the principal for whom the ticket-granting ticket should be obtained. If *principal* is omitted, the principal name from the existing cache is reused.

Description

The **kinit** command can be used to refresh a DCE credentials cache. When you invoke **kinit**, it prompts for your password.

The ticket lifetime and renewable lifetime are set in the following format:

{*num* {*interval*}}...

where:

num A number that specifies the number of the interval; *interval* can be specified by the following:

- **w** - weeks
- **d** - days
- **h** - hours
- **m** - minutes
- **s** - seconds

For example, to set the lifetime to 3 weeks, 5 days, and 10 hours, the entry would be the following:

3w5d10h

Files

/tmp/krb5cc_[*unix_id*] If the **KRB5CCNAME** environment variable is not set, the name of the file is in the form shown where [*unix_id*] is the decimal UNIX ID of the user. If the **KRB5CCNAME** environment variable is set, its setting determines the name of the file.

Related Information

Commands: **klist(1.sec), kdestroy(1sec).**

klist

Purpose

Lists cached tickets

Synopsis

klist [**-c** *cachename*] [**-e**] [**-f**]

Options

-c *cachename* Specifies that the contents of the cache identified by *cachename* should be displayed instead of the contents of the default cache.

-e Includes expired tickets in the display. Without this option, only current tickets are displayed.

-f Displays option settings on the tickets. The options are

- **D** (postdatable)
- **d** (postdated) **F** (forwardable)
- **f** (forwarded)
- **I** (initial)
- **i** (invalid)
- **P** (proxiable)
- **p** (proxy)
- **R** (renewable)

Description

The **klist** command lists the primary principal and tickets held in the default credentials cache, or in the cache identified by *cachename* if the **-c** option is used.

The name of the default credentials cache can vary between systems. However, if the **KRB5CCNAME** environment variable is set, its value is used to name the default cache. If it is not set, the form of the name is **/tmp/krb5cc_**[*unix_id*], where [*unix_id*] is the decimal UNIX ID of the user.

Related Information

Commands: **kinit(1sec)**, **kdestroy(1sec)**, **krb5(3)**.

su

Purpose Substitutes user ID temporarily

Synopsis **su [-K] [-f] [-l] [-m]** [*login*]

Options

-K Disables Kerberos authentication.

-f Prevents **csh(1)** from executing the **.cshrc** file.

-l Simulates a full login. The environment is discarded except for **HOME**, **SHELL**, **PATH**, **TERM**, and **USER**. **HOME** and **SHELL** are set to the target login's login shell and home directory. The **su** command changes the current directory to the target login's home directory. **USER** is set to the target login. **PATH** is set to **/bin:/usr/bin**. **TERM** is imported from your current environment. This option can not be specified with the **-m** option. They are mutually exclusive.

-m Leaves the environment unmodified. The invoked shell is your login shell, and no directory changes are made. As a security precaution, if the target user's shell is a nonstandard shell (as defined by **getusershell(3)**) and the caller's real UNIX ID is nonzero, **su** will fail. This option cannot be specified with the **-l** option. They are mutually exclusive.

Arguments

login The target user ID. If *login* is not specified, the root user ID is assumed. Note that only users in group 0 (normally **wheel**) can use **su** to become root.

By default (unless the prompt is reset by a startup file), the superuser prompt is set to # (number sign) to remind one of its awesome power.

Description

The **su** command lets you assume the identity of a different user. Note that the funtionality of the **su** command as described in this reference page may change, depending on the platform on which you are running the command.

The **su** command requests the password for *login* or, if *login* is not supplied, the password for root. If the correct password is supplied, **su** switches to that user and group ID after obtaining a ticket-granting ticket. A shell is then invoked with the real and effective person and group UNIX IDs of the new user. If authentication is disabled or not available, **su** uses the local **/etc/passwd** file for authentication.

If the **su** command is executed by root, no password is requested and a shell with the appropriate user ID is invoked; no additional Kerberos tickets are obtained.

Unless you specify the **-l** or **-m** option, when **su** is invoked the user environment is unmodified with the exception of the following:

- **HOME** and **SHELL** are set to the target login's default values. The current directory is changed to the target's login directory, and the target login's shell is invoked.
- **USER** is set to the target login value, unless the target login has a user ID of 0, in which case it is unmodified. The invoked shell is the target login's. This is the traditional behavior of **su**.

Related Information

Commands: **csh(1)**, **login(1)**, **sh(1)**, **kinit(1sec)**, **kerberos(1)**, **passwd(5)**, **group(5)**, **environ(7)**.

Chapter 10

DCE Distributed File Service

intro

Purpose

Introduction to the DFS commands

Description

DFS commands are divided into the following categories, or ''command suites'':

- The **bak** commands are issued by system administrators to operate the DFS Backup System.
- The **bos** commands are issued by users to list cell information; they are used by system administrators to use the Basic OverSeer Server (BOS Server).
- The **cm** commands are issued by users to determine machine, file, and directory information; they are used by system administrators to alter and configure the Cache Manager.
- The **fts** commands are issued by users to check quota information; they are used by system administrators to manipulate filesets.

DFS commands are divided into two groups: user-level commands and administrative-level commands. User-level commands are designated by the number 1 in the string **(1dfs)** following the command name; for example, **fts lsquota(1dfs)**. Administrative-level commands are designated by the number 8 in the string **(8dfs)** following the command name; for example, **fts create(8dfs)**.

All **bak** commands are administrative-level commands only. All administrative-level commands are documented in the *OSF DCE Administration Reference*.

DFS Command Structure

All DFS commands have the same general structure:

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

The following example illustrates the elements of a DFS command:

cm whereis [-path {*filename* | *directory_name*}...] **[-help]**

The following list summarizes the elements of a DFS command:

Command	A command consists of the command suite (**cm** in the previous example) and the command name (**whereis**). The command suite and the command name must be separated by a space. The command suite specifies the group of related commands to which the command belongs; it

indicates which program or server process executes the command. The command name directs the server process or program to perform an action.

Options	Command options always appear in bold type in the text, are always preceded by a **-** (dash), and are often followed by arguments. In the previous example, **-path** is an option, with *filename* or *directory_name* as its argument. An option and its arguments tell the server process or program which entities to manipulate when executing the command (for example, which file, which File Server machine, or which cell). In general, options should be specified in the order detailed in the documentation.
Arguments	Arguments for options always appear in italic type in the text. The **{ | }** (braces separated by a vertical bar) indicate that either one argument or the other (*filename* or *directory_name* in the preceding example) must be entered. The **...** (ellipsis) indicates that multiple arguments (*filename*s, *directory_name*s, or both) can be entered.
Optional information	Some commands have optional, as well as required, options and arguments. Optional information is enclosed in **[ ]** (brackets). All options and arguments in the previous example are optional.

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

Rules for Using DFS Commands

When supplying an argument with a command, the option associated with the argument can be omitted if

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax.
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.
- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

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

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

It is also valid to abbreviate a command name to the shortest form that still distinguishes it from the other command names in its suite. For example, it is acceptable to shorten the **fts help** command to **fts h** because no other command names in the **fts** command suite begin with the letter ''h.'' However, there are several **fts** commands that begin with ''l,'' such as **fts lsquota**, **fts lsmount**, and others. To avoid ambiguity, these commands can be abbreviated to **fts lsq** and **fts lsm**; other **fts** command names that begin with ''l'' can be abbreviated in a similar fashion.

The following examples illustrate three acceptable ways to enter the same **fts lsquota** command.

Complete command:

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

Abbreviated command name and abbreviated option:

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

Abbreviated command name and omitted option:

```
% fts lsq jlw/doc jlw/public
```

Privilege Required

No special privileges are required to execute user-level commands. Additional privileges required are included with individual commands.

Options

The following options are used with many commands. They are also listed with the commands that use them.

-cell	Specifies that the command is to be run with respect to the cell named by the *cellname* argument. By default, commands are executed in the local cell of the issuer of the command.
-path	Names the files, directories, or both to be used with the command.

-topic When used with **apropos** commands, specifies the keyword string for which to search. If it is more than a single word, surround it with " " (double quotes) or other delimiters. Type all strings in lowercase letters.

When used with **help** commands, specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **whereis**, not **cell whereis**). If this option is omitted, the command provides a short description of all commands in the suite.

-help Prints the online help for the command. All other valid options specified with this option are ignored.

Receiving Help

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

Reference pages for a command suite

To view the introductory page for a command suite, enter **man** followed by the command suite:

% man cm

Reference page for an individual command

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

% man cm_whereis

List of commands in a command suite

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

% cm help

Command syntax for a single command

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

% cm help whereis

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

The **apropos** command displays the first line of the online help entry for any command that has a specified string in its name or short description; this is useful if you cannot remember the exact name of a command. If the string is more than a single word, surround it with double quotes or other delimiters; type all strings in lowercase letters. For example, the following command produces a list of all **cm** commands with the word **status** in their names or descriptions:

```
% cm apropos -topic status
```

Cautions

Specific cautionary information is included with individual commands.

Related Information

Following is a list of all user-level commands. Administrative-level commands are not included here; they are described in the *OSF DCE Administration Reference*.

The **bos** Commands: **bos apropos(1dfs)**, **bos help(1dfs)**, **bos lscell(1dfs)**.

The **cm** Commands: **cm apropos(1dfs)**, **cm help(1dfs)**, **cm statservers(1dfs)**, **cm whereis(1dfs)**.

The **fts** Commands: **fts apropos(1dfs)**, **fts help(1dfs)**, **fts lsquota(1dfs)**.

bos apropos

Purpose Shows each help entry containing a specified string

Synopsis **bos apropos -topic** *string* [**-help**]

Options

-topic *string*
Specifies the keyword string for which to search. If it is more than a single word, surround it with "" (double quotes). Type all strings for **bos** commands in lowercase letters.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays the first line of the online help entry for any **bos** command containing the string specified by **-topic** in its name or short description.

Privilege Required

No privileges are required.

Output The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **bos** command where the string specified by **-topic** is part of the command name or the first line.

To see the syntax for a command, use the **bos help** command.

Examples The following command lists all **bos** commands that have the word **restart** in their names or short descriptions:

```
% bos apropos restart
getrestart: get restart times
restart: restart all processes
setrestart: set restart times for server processes
```

Related Information

Commands: **bos help(1dfs)**.

bos help

Purpose

Shows syntax of specified bos commands or lists functional descriptions of **bos** commands

Synopsis

bos help [-topic *string...*] **[-help]**

Options

-topic *string*
Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **status**, not **bos status**). If this option is omitted, the output provides a short description of all **bos** commands.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays the first line (name and short description) of the online help entry for every **bos** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **bos apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

Output

The online help entry for each **bos** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Examples

The following command displays the online help entry for the **bos status** command:

% bos help status

```
bos status: show server process status
Usage: bos status -server <machine> [-process <server_process>...]
[-long] [{-noauth | -localauth}] [-help]
```

Related Information

Commands: **bos apropos(1dfs).**

bos lscell

Purpose

Lists the cell in which the BOS Server is running

Synopsis

bos lscell -server *machine* [{**-noauth** | **-localauth**}] [**-help**]

Options

-server *machine*
: Names the server machine on which the BOS Server whose cell is to be listed is running. Specify the server name as a DCE pathname (for example, **/.../abc.com/hosts/fs1.abc.com**).

-noauth
: Directs **bos** to use the unprivileged identity **anonymous** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth
: Directs **bos** to use the identity (principal name) of the machine on which the command is issued as the identity of the issuer; this identity is not necessarily that of the machine on which the BOS Server is run. The issuer must be logged into the machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command lists the cell in which the BOS Server on the machine specified with the **-server** option is running. This information is taken from the local configuration file on the specified server machine.

Privilege Required

No privileges are required.

Output

This command displays the following line reporting the name of the cell in which the BOS Server is running:

```
Cell name is cellname
```

Examples

The following command displays the name of the cell in which the BOS Server on the machine named **fs1** is running:

```
% bos lscell /.../abc.com/hosts/fs1.abc.com
```

```
Cell name is abc.com
```

cm apropos

Purpose

Shows each help entry containing the specified string

Synopsis

cm apropos -topic *string* [**-help**]

Options

-topic *string*
: Specifies the keyword string for which to search. If it is more than a single word, surround the string with "" (double quotes) or other delimiters. Type all strings for **cm** commands in lowercase letters.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays the first line of the online help entry for any **cm** command containing the string specified by **-topic** in its name or short description.

Privilege Required

No privileges are required.

Output

The first line of an online help entry for a command names the command and briefly describes its function. This command displays that first line for any **cm** command where the string specified by **-topic** is part of the command name or first line.

To display the syntax for a command, use the **cm help** command.

Examples

The following command lists all **cm** commands that have the word **cache** in their names or short descriptions:

% cm apropos cache

```
flush: flush file data and status information from cache
getcacheinfo: get cache usage info
setcachesize: set cache size
```

Related Information

Commands: **cm help(1dfs)**.

cm help

Purpose

Shows syntax of specified cm commands or lists functional descriptions of **cm** commands

Synopsis

cm help [**-topic** *string...*] [**-help**]

Options

-topic *string*
Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **flush**, not **cm flush**). If this option is omitted, the output provides a short description of all **cm** commands.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays the first line (name and short description) of the online help entry for every **cm** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **cm apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

Output

The online help entry for each **cm** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Examples

The following command displays the online help entry for the **cm flush** command:

% cm help flush

```
cm flush: flush file from cache
Usage: cm flush [-path {<filename> | <directory_name>}...] [-help]
```

Related Information

Commands: **cm apropos(1dfs)**.

cm statservers

Purpose

Checks the statuses of File Server machines

Synopsis

cm statservers [{**-cell** *cellname* | **-all**}] [**-fast**] [**-help**]

Options

-cell *cellname*
: Specifies the name of the specific cell the Cache Manager is to probe for the status of each File Server machine it has contacted or has attempted to contact. The Cache Manager probes only the machines in the cell indicated with **-cell**. Use this option or use **-all**; omit both options for the Cache Manager to probe only the machines in the local cell.

-all
: Directs the Cache Manager to probe all of the machines it has contacted in all cells. Use this option or use **-cell**; omit both options for the Cache Manager to probe only the machines in the local cell.

-fast
: Directs the Cache Manager to display its current list of contacted File Server machines without probing the machines. This option can be combined with the **-cell** or **-all** option; it can also be used if both **-cell** and **-all** are omitted.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command lists all File Server machines in the indicated cells that meet the following two conditions:

- The Cache Manager has been in contact with the File Exporter running on that machine and needs to contact it in the future. (Reasons for wanting to contact a File Server machine include holding a token from that machine or having locked files on it.)

- The File Exporter on the machine is not currently responding to Cache Manager probes (implying that it is not responding to Cache Manager file requests either).

The Cache Manager maintains a list of File Server machines that meet the first condition, updating the list periodically by attempting to contact the File Exporter on each machine in the list. When a machine does not respond to the probe, the

Cache Manager marks it as nonfunctioning. If a machine that previously did not respond begins to respond again, the Cache Manager erases the mark.

This command used without the **-fast** option forces the Cache Manager to update its information immediately (rather than waiting the standard interval). The Cache Manager probes the File Exporter on the machines in the specified cell, records those that do not respond, and reports the result. If the issuer includes the **-fast** option, the Cache Manager outputs the list it already has at the time the command is issued, instead of probing the machines again.

By default, the Cache Manager probes machines in the local cell only. If the **-all** option is used, the Cache Manager probes all machines (from all cells) that meet the first condition. If a *cellname* is specified with the **-cell** option, the Cache Manager probes only the machines in that cell.

The execution of this command can be lengthy if a number of machines in the Cache Manager's list are unresponsive when the command is issued. The Cache Manager waits a standard time-out period before concluding that the File Exporter is not responding; this allows for the possibility of slow cross-network communication. If it is important that the command shell prompt return quickly, the issuer can run this command in the background. It is harmless to interrupt the command (with **<Ctrl-c>** or another interrupt signal).

This command does not check the status of all File Server machines in a cell. The Cache Manager probes only those machines that meet the first condition in the previous list.

Privilege Required

No privileges are required.

Output

If the Cache Manager gets a response from all of the machines that it probes (that is, all such machines are functioning normally), the output is `All servers are running`. This message does not imply that all File Server machines in the cells are running; it implies only that those machines the Cache Manager tried to probe are running.

If a machine fails to respond to the Cache Manager's probe within the time-out period, the output displays its name. This information comes from the kernel of the workstation where the command is issued.

Examples

1. In the following example, the issuer uses the **-fast** option to view the Cache Manager's current list of unresponsive machines belonging to the local cell rather than waiting for the Cache Manager to probe them again. The output indicates that all machines responded to the most recent probe:

   % cm statservers -f

   ```
   All servers are running.
   ```

2. The following command, run in the background, checks File Server machines the Cache Manager has previously contacted in all cells. It reports that the machines **fs1** and **fs3** did not respond to the Cache Manager's probes. The **&** (ampersand) is used to execute the command in the background.

   % cm statservers -all &

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

cm whereis

Purpose

Reports names of File Server machines housing specified files or directories

Synopsis

cm whereis [-path {*filename* | *directory_name*}...] **[-help]**

Options

-path *filename* or *directory_name*
: Specifies the pathname of each file or directory whose location is to be reported. Each file or directory must reside in DFS, not on a local disk. If a full pathname is not provided, the file or directory is assumed to reside in the current working directory. If this option is omitted, the current working directory is used.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command returns the names of the File Server machines that house the filesets containing each file or directory indicated with the **-path** option. This information comes from the kernel of the workstation on which the command is issued.

Privilege Required

No privileges are required.

Output

The output includes a separate line displaying the following information about each file or directory specified with the **-path** option:

```
File 'filename' resides in the cell 'cellname' in fileset
'fileset_name' on host(s) 'machine_name(s)'.
```

- The *filename* in the output specifies the name of a file or directory specified with the **-path** option.
- The *cellname* in the output specifies the name of the cell in which the file or directory is located.
- The *fileset_name* in the output specifies the name of the fileset on which the file or directory is located.

- The *machine_name(s)* in the output specifies the name of one or more File Server machines on which the fileset is located. If the fileset is a read/write fileset, only one machine name is displayed; if the fileset is a read-only fileset, multiple machine names can be displayed.

Examples

The following command indicates that the directory **/.../abc.com/fs/bin/sysfile** is located in a replicated fileset on the File Server machines named **fs1**, **fs3**, and **fs6**, all of which are located in the cell named **abc.com**:

% cm whereis /.../abc.com/fs/bin/sysfile

```
File '/.../abc.com/fs/bin/sysfile' resides in the cell 'abc.com',
in fileset 'sysfile.bin', on hosts fs1.abc.com, fs3.abc.com, fs6.abc.com.
```

fts apropos

Purpose

Shows each help entry containing a specified string

Synopsis

fts apropos -topic *string* [**-help**]

Options

-topic *string*
Specifies the keyword string for which to search. If it is more than a single word, surround it with " " (double quotes) or other delimiters. Type all strings for **fts** commands in all lowercase letters.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays the first line of the online help entry for any **fts** command containing the string specified by **-topic** in its name or short description.

Privilege Required

No privileges are required.

Output

The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **fts** command where the string specified by **-topic** is part of the command name or first line.

To see the syntax for a command, use the **fts help** command.

Examples

The following command lists all **fts** commands that have the word **mount** in their names or short descriptions:

```
% fts apropos mount
```

```
crmount: make mount point
delmount: remove mount point
lsmount: list mount point
```

Related Information

Commands: **fts help(1dfs)**.

fts help

Purpose

Shows syntax of specified fts commands or lists functional descriptions of **fts** commands

Synopsis

fts help [**-topic** *string*...] [**-help**]

Options

-topic *string*
Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **lsft**, not **fts lsft**). If this option is omitted, the output provides a short description of all **fts** commands.

-help
Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays the first line (name and short description) of the online help entry for every **fts** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

Use the **fts apropos** command to show each help entry containing a specified string.

Privilege Required

No privileges are required.

Output

The online help entry for each **fts** command consists of the following two lines:

- The first line names the command and briefly describes its function.
- The second line, which begins with `Usage:`, lists the command options in the prescribed order.

Examples

The following command displays the online help entry for the **fts delmount** command:

% fts help delmount

```
fts delmount: remove mount point
Usage: fts delmount -dir <directory_name>...  [-help]
```

Related Information

Commands: **fts apropos(1dfs)**.

fts lsquota

Purpose Shows quota and usage information about filesets and aggregates

Synopsis **fts lsquota** [{**-path** {*filename* | *directory_name*}... | **-fileset** {*name* | *ID*}...}] [**-cell** *cellname*] [{**-noauth** | **-localauth**}] [**-verbose**] [**-help**]

Options

-path *filename* or *directory_name*
Names a file or directory from each fileset about which quota and usage information is to be displayed. Include filenames or directory names from different filesets if desired. It is not necessary to name more than one file or directory from the same fileset. Use this option or use **-fileset**; omit both options to display information about the fileset containing the current working directory.

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

-cell *cellname*
Specifies the cell with respect to which the command is to be run. The default is the local cell of the issuer of the command.

-noauth Directs **fts** to use the unprivileged identity **anonymous** as the identity of the issuer of the command. If you use this option, do not use the **-localauth** option.

-localauth Directs **fts** to use the identity (principal name) of the machine on which the command is issued as the identity of the issuer. The issuer must be logged into the machine as **root** for this option to work. If you use this option, do not use the **-noauth** option.

-verbose Directs **fts** to provide detailed information about its actions during command execution.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Description

This command displays quota and usage information about filesets and the partitions or aggregates on which they reside. Use the **-path** option to specify a file or directory on a fileset to see information about that fileset; use the **-fileset** option to specify the name or ID number of a fileset to see information about that fileset; omit both options to see information about the fileset that contains the current working directory.

For DCE LFS filesets, the **fts lsquota** command displays the maximum quota, the number of kilobytes in use, and the percentage of the quota in use. For both DCE LFS and non-LFS filesets, this command displays the name of the fileset, information about the number of kilobytes on the aggregate or partition on which the fileset resides, the number of kilobytes in use on the aggregate or partition, and the percentage of the aggregate or partition in use. It also reports whether the device is a DCE LFS aggregate or a non-LFS partition.

The size of a non-LFS fileset is equal to the size of the partition on which it resides. Therefore, the quota and usage information displayed for the partition (non-LFS aggregate) in the output of the **fts lsquota** command equals the quota and usage information of the fileset on the partition. Using this command with a non-LFS fileset is the equivalent of using the UNIX **df** command with the partition on which the fileset resides. (Note that the **df** command cannot be used to display the size of a DCE LFS aggregate or fileset.)

The **fts lsheader** and **fts lsft** commands can be used to display the maximum quota of a DCE LFS fileset. The **fts aggrinfo** command can be used to display the total disk space on an aggregate and the amount currently available.

By default, every newly created DCE LFS fileset has a quota of 5000 kilobytes. The **fts setquota** command can be used to increase or decrease the quota of a DCE LFS fileset. Because the quota of a DCE LFS fileset does not represent the amount of physical data stored on the fileset, it can be larger than the size of the aggregate on which the fileset resides. Similarly, the combined quotas of all filesets on an aggregate can be larger than the size of the aggregate.

The quota of a non-LFS fileset cannot be changed via DFS.

Privilege Required

No privileges are required.

Output

This command displays the following information about each specified fileset:

- The name of the fileset
- The quota, in kilobytes, of the fileset (DCE LFS only)

- The number of kilobytes currently in use on the fileset (DCE LFS only)
- The percentage of the quota currently in use on the fileset (DCE LFS only)
- The percentage of available disk space currently in use on the aggregate on which the fileset resides
- The number of kilobytes in use on the aggregate and the total number of kilobytes on the aggregate
- The file system type of the aggregate (`LFS` or `non-LFS`)

If the fileset usage rises above 90% or the aggregate usage rises above 97%, the appropriate percentage is indicated with `<<` and the message `<<WARNING` is displayed after the aggregate usage information on that line of the output.

Note: Because each non-LFS aggregate (partition) contains a single fileset, the information displayed for a non-LFS aggregate applies to the single non-LFS fileset it houses. Ignore the quota, usage, and percentage values displayed for a non-LFS fileset; they are always 0 (zeros). Consult the quota, usage, and percentage values displayed for the partition on which the non-LFS fileset resides to determine the corresponding values for the fileset.

Examples

1. The command that follows lists quota and usage information for the filesets and aggregates that contain the directories named **/.../abc.com/fs/usr/terry** and **/.../abc.com/fs/usr/jlw**. The first directory resides on the DCE LFS fileset named **user.terry**; the quota of the DCE LFS fileset is less than the quota of the aggregate on which it is located. The second directory resides on the non-LFS fileset named **user.jlw**; the quota of the non-LFS fileset is the same as the quota of the partition on which it is located.

% fts lsq /.../abc.com/fs/usr/terry /.../abc.com/fs/usr/jlw

```
Fileset Name   Quota   Used   % Used   Aggregate
user.terry     15000   5071      34%     86% = 84538/98300 (LFS)
user.jlw           0      0       0%     84% = 8448/10000 (non-LFS)
```

2. The following command lists quota and usage information for the DCE LFS fileset named **user.jean** and the aggregate on which it resides. The `<<WARNING` message directs the issuer's attention to the fact that the percentage of the quota in use on the indicated fileset is well above the warning level of 90%.

% fts lsq -f user.jean

```
Fileset Name   Quota   Used   % Used   Aggregate
user.jean       5000   4955     99%<<   92% = 87436/98300 (LFS) <<WARNING
```

Related Information

Commands: **fts aggrinfo(8dfs)**, **fts lsft(8dfs)**, **fts lsheader(8dfs)**, **fts setquota(8dfs)**.

Index

OPEN SOFTWARE FOUNDATION™

INFORMATION REQUEST FORM

Please send to me the following:

( ) OSF™ Membership Information

( ) OSF™DCE License Materials

( ) OSF™DCE Training Information

Contact Name ______________________

Company Name ______________________

Street Address ______________________

Mail Stop ______________________

City ____________ State ______ Zip ________

Phone ____________ FAX ____________

Electronic Mail ______________________

MAIL TO:

Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142

Attn: OSF™DCE

For more information about OSF™DCE call OSF Direct Channels at 617 621 7300.

OSF™DCE

OSF™DCE User's Guide and Reference

Printed in the U.S.A.

Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142

Prentice Hall, Inc.

ISBN 0-13-643842-3

90000

9 780136 438427

DISTRIBUTED SYSTEMS