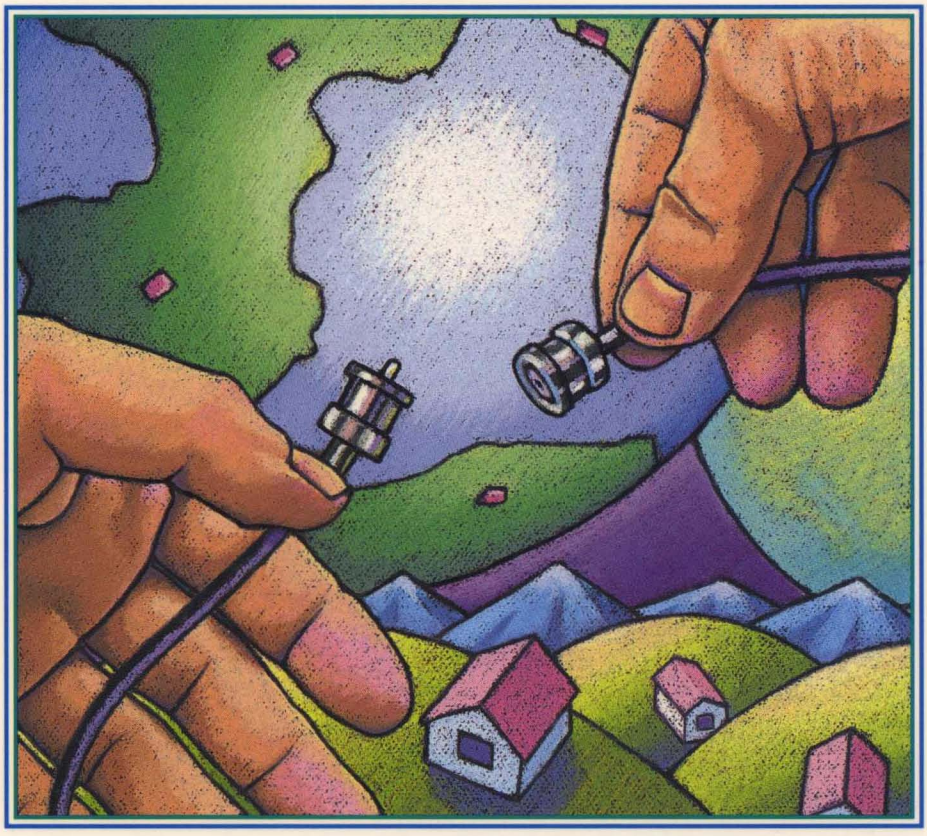
OSF™DCE

OSF™DCE Administration Guide — Extended Services



OPEN SOFTWARE FOUNDATION

OSF® DCE Administration Guide — Extended Services

Revision 1.0

Open Software Foundation

P T R Prentice Hall, Englewood Cliffs, New Jersey 07632

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

This book was formatted with troff

Published by P T R Prentice Hall
Prentice-Hall, Inc.
A Paramount Communications Company
Englewood Cliffs, New Jersey 07632

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

ISBN 0-13-176561-2

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 3. DCE Diskless Support Service

List of Figures

List of Tables

Preface

The *OSF DCE Administration Guide* provides concepts and procedures that enable you to manage the Distributed Computing Environment (DCE). Basic DCE terms are introduced throughout the *OSF DCE Administration Guide*. A glossary for all of the DCE documentation is provided in the *Introduction to OSF DCE*. The *Introduction to OSF DCE* helps you to gain a high-level understanding of the DCE technologies and describes the documentation set that supports DCE.

Audience

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

Applicability

This is Revision 1.0 of this guide. It applies to the OSF® DCE Version 1.0 offering and related updates. (See your software license for details.)

Purpose

The purpose of this guide is to help system and network administrators to plan, configure, and manage DCE. After reading the guide, you will understand what the system administrator needs to do to plan for DCE. Once you have built the DCE source code on your system, use this guide to assist you in installing executable files and configuring DCE. The *OSF DCE Release Notes* contain instructions for installing and building DCE source code.

Document Usage

The *OSF DCE Administration Guide* consists of three books, each of which is divided into parts, as follows:

- *OSF DCE Administration Guide—Introduction*
  - — Part 1. Introduction to DCE Administration
  - — Part 2. Configuring and Starting Up DCE
- *OSF DCE Administration Guide—Core Components*
  - — Part 1. DCE Remote Procedure Call
  - — Part 2. DCE Cell Directory Service
  - — Part 3. DCE Distributed Time Service
  - — Part 4. DCE Security Service
- *OSF DCE Administration Guide—Extended Services*
  - — Part 1. DCE Global Directory Service

— Part 2. DCE Distributed File Service

— Part 3. DCE Diskless Support Service

Related Documents

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

- *Introduction to OSF DCE*
- *OSF DCE Administration Reference*
- *OSF DCE User's Guide and Reference*
- *OSF DCE Release Notes*
- *OSF DCE Porting and Testing Guide*
- *OSF DCE Application Development Guide*
- *OSF DCE Application Development Reference*

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—Introduction* and the *OSF DCE Release Notes*.

Part 1

Global Directory Service

Chapter 1

Overview of the X.500 Directory Service

The Global Directory Service (GDS) is a distributed, replicated directory service. ''Distributed'' in this context means that information is stored in different places in the network. Requests for information are routed by GDS to directory servers throughout the network. ''Replicated'' in this context means that information is stored in more than one location for easier access.

GDS is based on the CCITT X.500/IS 9594 (1988) international standard. The aim of this standard, also referred to as the International Organization for Standardization (ISO) directory standard, is to provide a global directory that supports network users and applications with information required for communication. The directory plays a significant role in allowing the interconnection of information processing systems from different manufacturers, under different managements, of different levels of complexity, and of different ages.

GDS is the DCE implementation of the OSI directory standard. Together with the Cell Directory Service (CDS), it provides its users with a centralized place to store information required for communication, which can be retrieved from anywhere in a distributed system. GDS maintains information describing objects such as people, organizations, applications, distribution lists, network hardware, and other distributed services dispersed over a large geographical area.

CDS stores names and attributes of resources located in a DCE cell. A DCE cell consists of various combinations of DCE machines connected by a network. Each DCE cell contains its own CDS Server, which provides access to local resource information. CDS is optimized to provide users with access to local information.

GDS serves as a general-purpose information repository. It provides information about resources outside a DCE cell. GDS links the various cells by helping to locate remote cells.

1.1 The Directory Information Model

A *directory* is a collection of information about objects that exist in the world. Objects are anything with a name; for example, people, organizations, printer servers, and application processes.

All types of information can be stored in a directory. This information is usually in the form of a description or an address. For example, a typical directory is a white pages city telephone directory containing information about people and businesses. The addressing information stored in the directory consists of the telephone number and street address of a person or business. Descriptive information is the name of the person or business. Another example is a yellow pages directory that provides information about an object, such as a restaurant. A specific restaurant may include additional information such as types of food, serving times, credit cards accepted, and so on.

The information stored in a directory is known as the *Directory Information Base* (DIB).

1.1.1 Entries

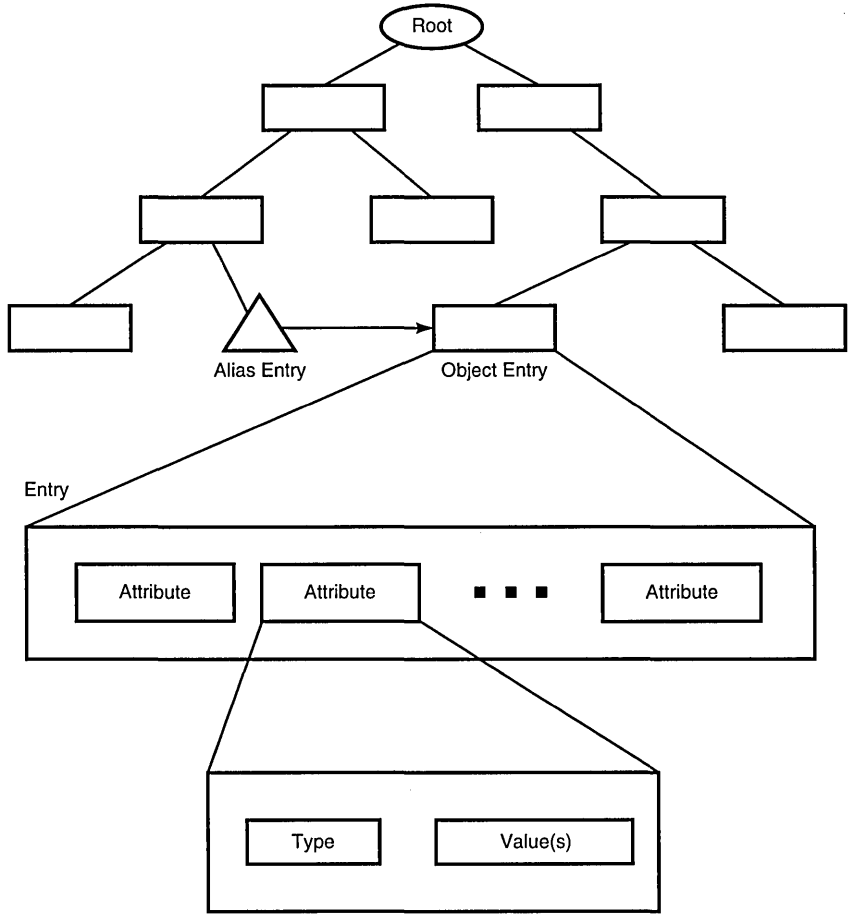
Information about an object is stored in an entry in the DIB. There are two types of entries in the directory. An *object* entry refers to an object and contains all the information about that object in the DIB. An *alias* entry is a special type of entry that provides an alternative name for an entry. Alias entries are described later in this chapter.

1.1.2 Attributes

Each entry in the DIB is made up of a set of attributes. Each attribute stores information about the object to which the entry refers. An entry for a person can contain separate attributes for that person's last name, first name, postal address, telephone address, and so on.

Each attribute, in turn, is made up of an *attribute type* and one or more *attribute values*. The attribute type identifies the attribute, and the attribute value is the value or set of values for that attribute. For the attributes for an entry for a person, the attribute types would be **Surname**, **Common-Name**, **Postal-Address** and **Telephone-Number**. Attributes that have more than one value are called *multivalued* attributes. For example, the entry for someone who has more than one telephone number would contain one **Telephone-Number** attribute with two values, one for each number. Figure 1-1 shows the structure of the DIB.

Figure 1–1. The Structure of the DIB



Each attribute type also has an *attribute syntax* which describes the format of the legal values of that attribute type. For example, the **Telephone-Number** attribute type is declared to have values of attribute syntax **Telephone Number Syntax**. This means that a value takes the form of a valid telephone number (irrespective of whether or not it is a telephone number). Another example is **Common-Name**, which is assigned the **Case Ignore String** syntax so that only strings can be entered, and their case is not significant.

1.1.3 Object Classes

In order to categorize entries in the DIB, each entry is said to contain one or more *object classes*. The object class of an entry describes the object that entry refers to. Sample object classes are **Country**, **Person**, **Organizational-Person**, and **Application-Entity**. The object class of an entry restricts the permitted attributes for that entry. The mandatory and optional attributes of entries in an object class are determined by *object class rules*. (These rules are part of a schema and are described later in this chapter.)

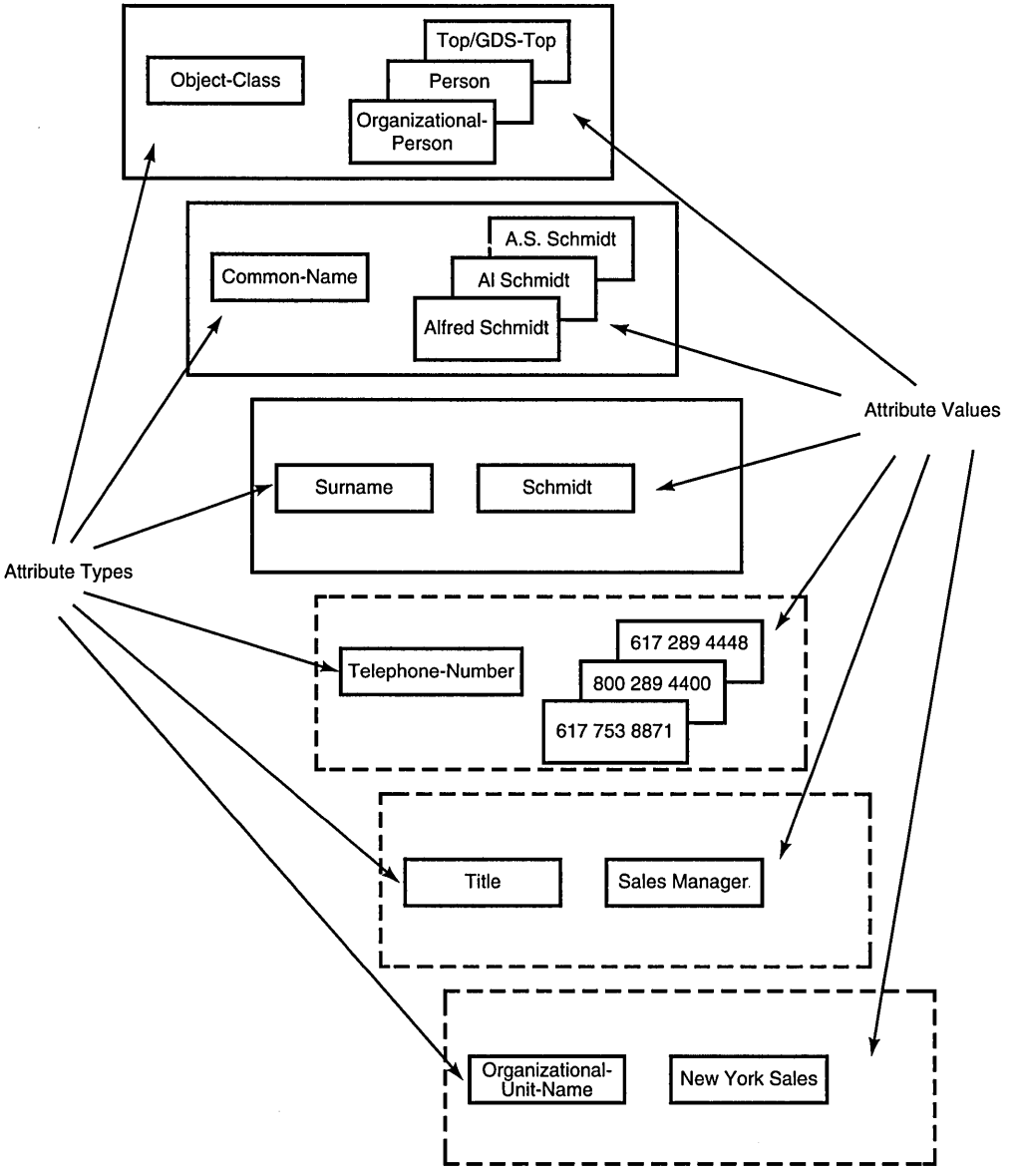
For example, an entry representing an organization must contain an attribute called **Organization-Name**, which has the name of the organization as its value. It can contain optional attributes that describe the organization, the state or locality in which the organization resides, the postal address of the organization, the business category of the organization, and so on.

As a general rule, *all* entries must contain the **Object-Class** attribute, which contains the list of object classes to which the entry belongs. This attribute is multivalued. If an entry belongs to more than one object class, all object classes must be listed in this attribute.

For example, there are two object classes defined as **Person** and **Organizational-Person**. An entry in the **Person** object class must contain a **Common-Name** attribute and a **Surname** attribute. It can contain several other attributes such as **Description**, **Telephone-Number**, and **User-Password**. An entry in the object class **Organizational-Person** is defined as a subclass of **Person**. This means that it must contain every mandatory attribute of **Person** and can contain every optional attribute of **Person**. However, it is also defined as an extension of **Person**. **Organizational-Person** can contain more attributes than **Person**. These attributes include **Organizational-Unit-Name**, **Telex-Number**, and so on. An entry that describes **Organizational-Person** must have at least two values in its **Object-Class** attribute, that is **Person** and **Organizational-Person**.

Figure 1-2 shows an example of an entry describing **Organizational-Person**.

Figure 1–2. Example of an Entry Describing Organizational Person



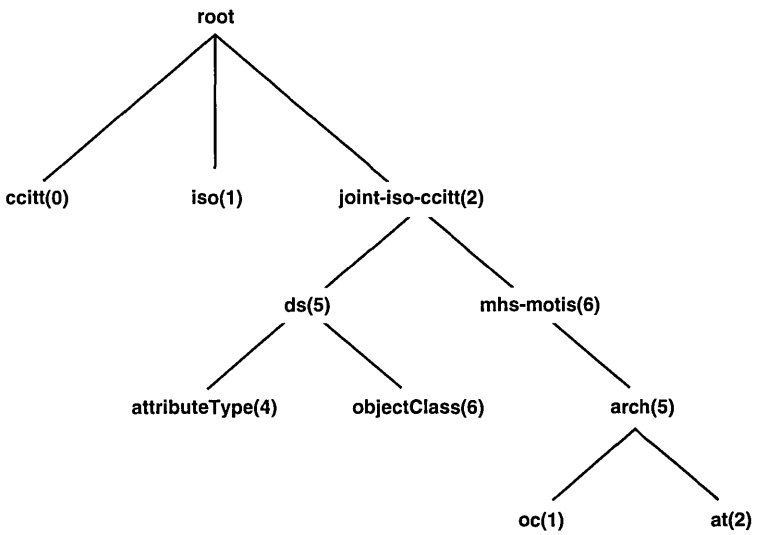
1.1.4 Object Identifiers

As shown in the previous section, attribute types and object classes have human-language names. Although all names are meaningful and unique, they are not used in the protocols; an Object Identifier (OID) is used instead. This is a hierarchical number assigned by a registration authority. The possible values of OIDs are defined in a tree. The standards bodies ISO and CCITT control the top of the tree and define portions of this tree. These standards bodies delegate the remaining portions to other organizations so that each object class, attribute type, and attribute syntax has a unique OID. These OIDs are written in the format **2.5.6.2**; this is the OID of the **Country** object class. If more information is required, the OIDs can be written as follows:

joint-iso-ccitt {2} modules {5} object classes {6} country {2}

Figure 1-3 shows a portion of the tree.

Figure 1–3. Object Identifiers



To enable more efficient transmission over a communications line, the first two subidentifiers **2** and **5** are coded together; this results in one subidentifier, **85**. The encoded format of the OID for the **Country** object

class is thus **85.6.2**. This encoded representation is used in the GDS administration programs and in tables in this book. For example, Table 1-5 (shown later in this chapter), which contains a list of OIDs for selected object classes, uses this format. Where unencoded format is used in text, the encoded format is included in parentheses for clarity.

1.2 X.500 Naming Concepts

Large amounts of information need to be organized to enable efficient data retrieval and ensure that names are unique. Information in the DIB is organized in a hierarchical structure called the *Directory Information Tree* (DIT). The structure and naming of the nodes in the DIT are specified by registration authorities for a standardized set of X.500 names, and by implementors of the directory service (such as OSF) for implementation-specific names. A schema describes the DIT hierarchy. Schemas are described in more detail later in Section 1.6.

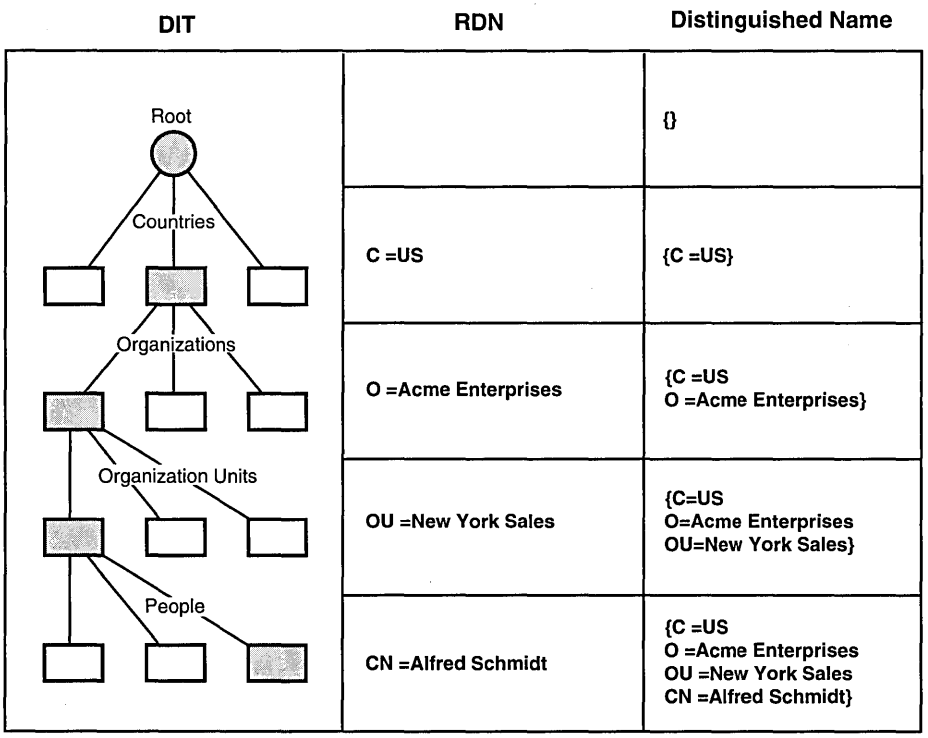
Although the X.500 standard does not define a specific schema, it does make general recommendations. For example, countries and organizations need to be named near the root of the DIT; whereas, people, applications, and devices need to be named further down in the hierarchy. GDS supplies a default schema that complies with these recommendations.

1.2.1 Distinguished Names

A hierarchical path exists from the root of the DIT to any entry in the DIB. Each entry must have a name that uniquely describes that entry. A *Relative Distinguished Name* (RDN) distinguishes an entry from other entries with the same superior node in the DIT. A sequence of RDNs, starting from the root of the tree, can identify a unique path down the tree, and thus a unique entry. This sequence of RDNs, which identifies a particular entry, is the *Distinguished Name* (DN) of that entry. Each entry in the DIB can be referenced by its DN.

Figure 1-4 shows an example of a DN. The shaded boxes in the DIT represent the entries that are named in the column labeled RDN. According to the schema, countries are named directly below the root, followed by organizations, organization units, and people.

Figure 1–4. A Distinguished Name in a Directory Information Tree



Every entry in the DIB has a DN, not just the leaf nodes. For example, the DN for Acme Enterprises in Figure 1-4 is the concatenation of the DN of the entry above with its RDN.

1.2.2 RDNs and Attribute Value Assertions

An RDN consists of one or more assertions about the type and value of an attribute. A pair consisting of an attribute type and a value of that type is known as an *Attribute Value Assertion* (AVA). All attribute types in an RDN must be different. The value of an attribute in an RDN's AVA is called the

distinguished value of that attribute as opposed to the other possible values of that attribute.

For example, the entry shown in Figure 1-4 contains the RDN **CN = Alfred Schmidt**. Although the **CN (Common-Name)** attribute consists of three values: Alfred Schmidt, A. S. Schmidt, and Al Schmidt, the AVA **CN = Alfred Schmidt** contains the value **Alfred Schmidt**, which has been designated as the distinguished value in the AVA.

An RDN usually contains a single distinguished value, and therefore is made up of a single AVA. However, under certain circumstances additional values (and hence multiple AVAs) are used.

For example, the RDN of an **Organizational-Person** entry is usually composed of a single AVA, such as the **Common-Name** attribute type with a distinguished value (in Figure 1-4, the AVA is **CN = Alfred Schmidt**). Depending on the schema, the RDN of an **Organizational-Person** entry may contain more than one AVA. For example, the RDN in Figure 1-4 can contain the AVAs **CN = Alfred Schmidt** and **OU = New York Sales**, with **Alfred Schmidt** and **New York Sales** as distinguished values.

To summarize:

- A DIT consists of a collection of DNs.
- DNs result from a concatenation of RDNs.
- RDNs consist of an unordered collection of attribute type and value pairs (AVAs).

1.2.3 Aliases

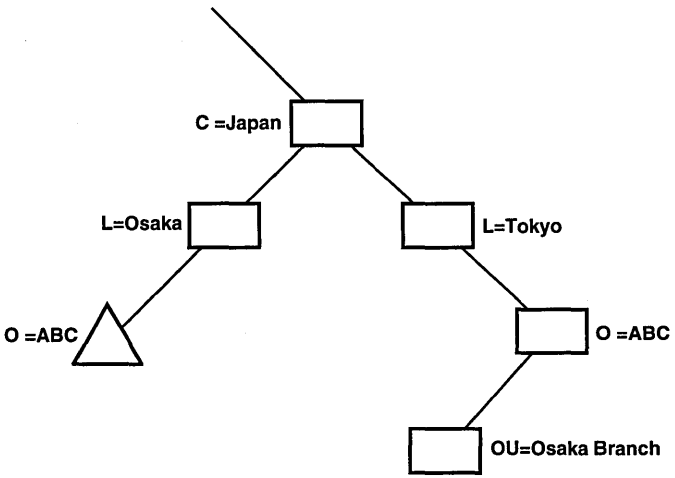
Alternative names or aliases are supported in the DIT through special pointer entries called *alias entries*. Alias entries do not contain any attributes other than their distinguished attributes, the object class attribute, and the aliased object name attribute (that is, the DN of the aliased object entry). Because an alias entry has no subordinate entries, it is, by definition, a leaf entry of the DIT as shown in Figure 1-5. Alias entries point to object entries and provide the basis for alternative names for the corresponding objects.

For example, aliases are used to provide more user-friendly names, to direct the search for a particular entry, to reduce the scope of a search, to provide

for common alternative abbreviations and spellings, or to provide continuity after a name change.

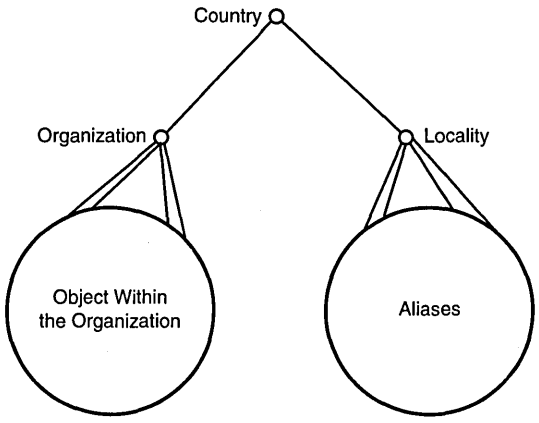
Figure 1-5 demonstrates how an alias name provides continuity after a name change. The **ABC** company's branch office, located originally in Osaka, has moved to Tokyo. To make the transition easier for directory users and to guarantee that a search based on the old information finds its target, an alias for **O=ABC** is added to the directory beneath **L=Osaka**. This alias entry points to the object entry **O=ABC**. A search for **ABC** under **L=Osaka** in the DIT finds the entry **/C=Japan/L=Tokyo/O=ABC**.

Figure 1–5. An Alias in the Directory Information Tree



Alias entries can also be used instead of *filtering*. Filtering is a process that uses assertions about particular attributes to search through the DIT. Although filtering does not require any special information in the DIT, a search involving a large population of entries and attributes can be expensive. An alternative approach is to set up special subtrees whose naming structures are designed to perform searches similar to yellow pages searching. Figure 1-6 shows an example of such a subtree, populated by alias entries and segregated by localities within the organization. Note that the entries within these special subtrees could also be a mixture of object and alias entries, provided there is only one object entry for each object stored in the directory.

Figure 1–6. A Subtree Populated by Aliases



An object with an entry in the DIT can have zero or more aliases. Several alias entries can point to the same object entry. An alias entry can point to an object that is not a leaf entry. Only object entries can have aliases; aliases of aliases are not permitted.

1.3 GDS as a Distributed Service

GDS has two basic functions in a DCE cell:

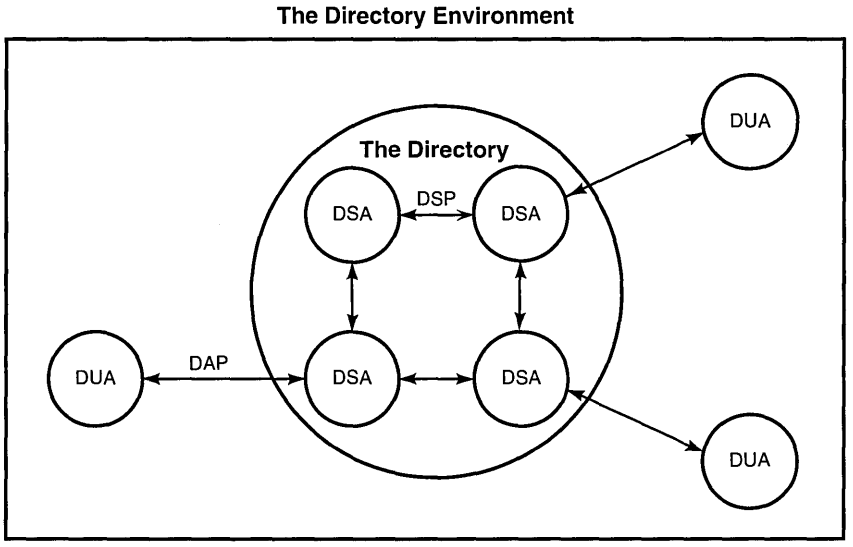
- It provides a high-level, worldwide directory service by tying together independent DCE cells.
- It is used as an additional directory service to CDS for storing object names and attributes in a central place.

The GDS database contains information that can be distributed over several GDS servers. In addition, copies of information can be stored in multiple GDS servers, and the information can be cached locally. The unit of replication in GDS is the entry (whole subtrees can also be replicated).

The information belonging to the DIB is shared between several *Directory Service Agents* (DSAs). A DSA is a process that runs on a GDS server machine and manages the GDS database. DSAs that know only part of the total directory information (as shown in Figure 1-7) cooperate with each

other to perform directory service operations. This cooperation often involves the navigation of operations through the network.

Figure 1–7. The Relationship Between the DSA and the DSU



Users access the directory through *Directory User Agents* (DUAs). DUAs issue requests to DSAs on behalf of users requesting directory service operations. The manner in which DUAs talk to DSAs is defined by the X.500 standard.

The *Directory Access Protocol* (DAP) is defined for communication between DUAs and DSAs.

The directory standard also defines directory functions in the DAP. The directory functions can be divided into three general categories:

- *Read* operations involve the retrieval of information from specific named entries. This enables name-to-attribute mapping that is similar to the white pages telephone directory.
- *Search* operations involve general browsing and relational searching of information. They support human interaction with the directory and are similar to the yellow pages telephone directory.

- *Modify* operations are used to modify the information in the directory.

The *Directory System Protocol* (DSP) is defined by the directory standard to allow DSAs to communicate with one another. DSP provides three methods of distributed request resolution: referral, chaining, and multicasting.

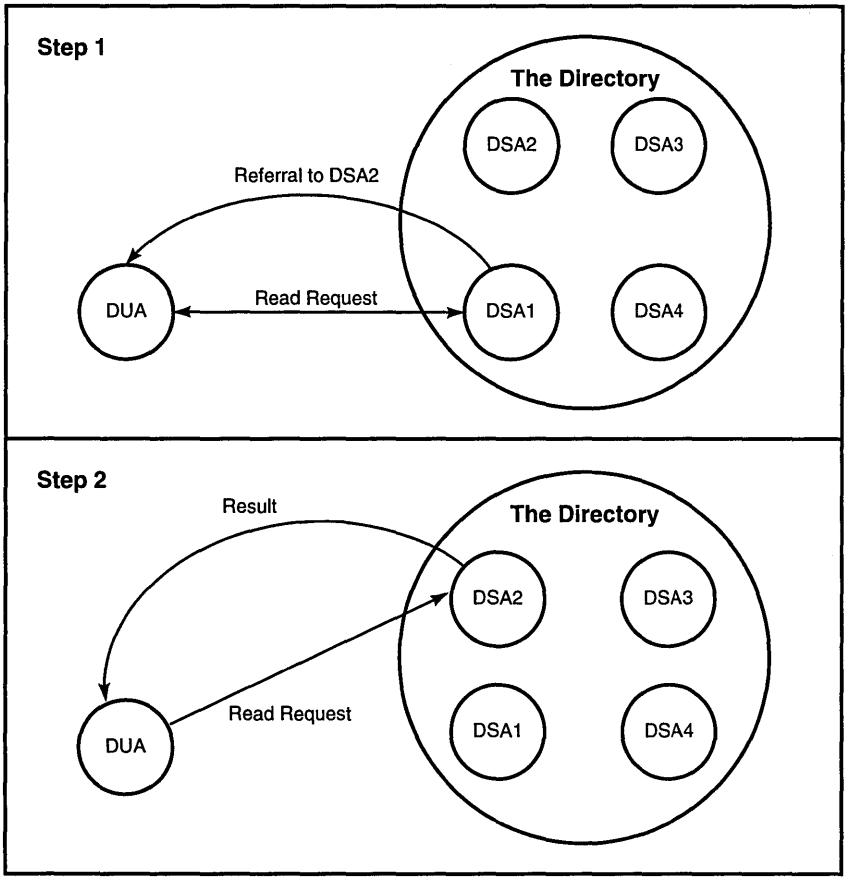
1.3.1 Referral

In some cases, a DSA is unable to provide a service to a DUA because the required information is held elsewhere in the network. Using a method called *referral*, a DSA informs the DUA or the calling DSA where the information is located. The referral method may be initiated by the user's preference or the DSA's circumstances.

Referrals are possible because the DN provided by the DUA identifies where the requested entry is located in the DIT. DSAs use their knowledge of the DIT to inform the DUA of the DSA that holds the requested information (or a DSA that is closer to the DSA holding the information).

Figure 1-8 shows an example of referral. **DSA1** passes a referral to **DSA2** back to **DUA. DUA** then makes a request to **DSA2.**

Figure 1–8. An Example of a Referral



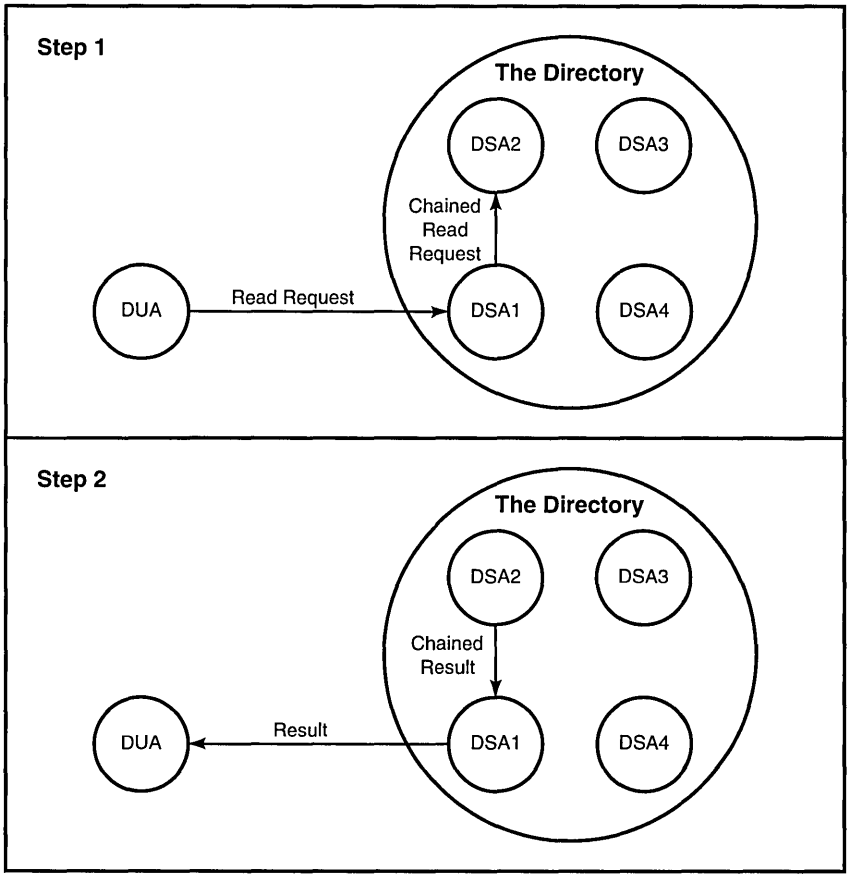
1.3.2 Chaining

If a request received from a DUA cannot be fulfilled by the receiving DSA, that DSA sends a referral back to the initiating DUA over DAP. Alternatively, the DSA chains the request over DSP, asking another DSA to perform the requested function. That DSA either performs the function or sends back a referral of its own. In either case, the first DSA eventually responds to the originating DUA with the results of the completed operation or a referral. This process is called *chaining*.

Chaining can go deeper than one level. To prevent lengthy searches, a user needs to specify no chaining or a limit on total elapsed time for an operation.

Figure 1-9 shows an example of chaining. **DUA** issues a request to **DSA1**. **DSA1** is unable to service the request and passes it to **DSA2**. **DSA2** services the request, passes the result back to **DSA1**, and **DSA1** passes the result back to **DUA**.

Figure 1–9. An Example of Chaining



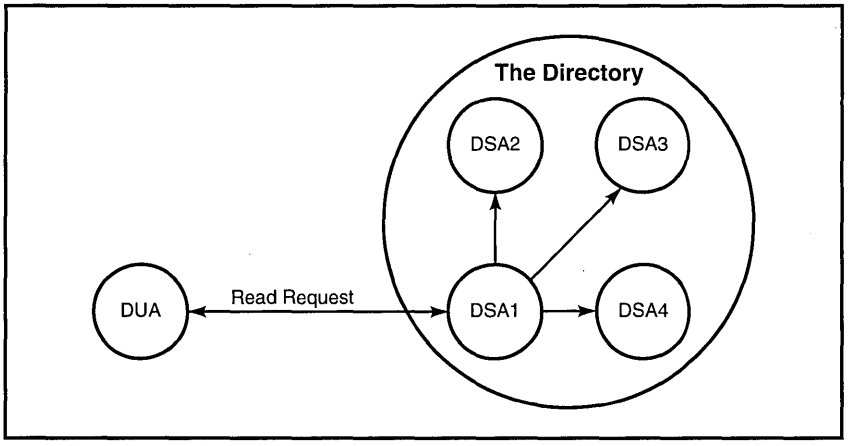
1.3.3 Multicasting

A DSA can forward a request to two or more DSAs simultaneously. This process is called *multicasting*. The multicasting DSA waits for responses from the DSAs to which it forwarded the request, then builds and sends a single response back to the requestor.

Multicasting is only used in the unlikely event of name resolution, when a DSA is tracking nonspecific subordinate references.

Figure 1-10 shows an example of multicasting. **DUA** issues a request to **DSA1**. **DSA1** carries out the request by forwarding it to **DSA2**, **DSA3**, and **DSA4**.

Figure 1–10. An Example of Multicasting



1.3.4 Directory Distribution

The DIB is potentially distributed transparently across multiple DSAs, with each DSA comprising part of the DIB. Conceptually, DSAs are made up of two types of information: *Directory Information* is the collection of entries over which the administrator of a particular DSA has administrative authority; the information required to locate the DSA with administrative authority for a particular entry is *Knowledge Information*.

In the GDS implementation, every object entry has an attribute called the **Master Knowledge** attribute. This attribute contains the DN of the DSA

responsible for maintaining the "master copy" of the data. The entry can be replicated and stored on other DSAs. The DSA that maintains the master copy updates the entry periodically in a process called *shadowing*. The frequency of these updates is determined by the administrator through shadow administration functions (described in Chapter 10). The updated entries distributed on one or more DSAs are "shadow copies" of the entry that masters the object.

Each DSA has an entry containing its own name and a network address. Each DSA knows its own name from a configuration file. By comparing this name with the **Master Knowledge** attribute of an entry in the DSA, the DSA determines whether it masters the entry or contains a shadow copy of the entry.

Figure 1-11, presented later in this chapter, shows an example of how master and shadow entries are stored on two DSAs: **DSA1** and **DSA2**. The figure shows the copy of **AP2** on **DSA1** with an indication that the **Master Knowledge** attribute is set to the DN of **DSA2**. **AP2** is mastered on **DSA2**.

A master entry can be modified only by an administrator with the appropriate access rights to the entry.

1.3.5 Operations Standardized by X.500

Table 1-1 lists and briefly describes the underlying operations defined by X.500 that are used internally by GDS to provide the functions available in the GDS administration programs. These operations are available to programmers in the library of XDS API function calls that are included with GDS. XDS API allows programmers to write applications that perform operations on the directory. (See the *OSF DCE Application Development Guide* for more information on programming with XDS API.)

Table 1–1. Operations Standardized by X.500

Operation	Meaning
Bind	Used at the beginning of a directory session for accessing the directory
Unbind	Used at the end of a directory session to release the directory association
Read	Reads a particular entry with particular attributes
Compare	Checks whether an attribute value supplied matches a value of that attribute of a particular entry
List	Lists all the immediate subordinates of a particular entry
Search	Returns information from all the entries (that satisfy some filter) within a certain section of the DIT
Abandon	Cancels an outstanding interrogation request
Add entry	Adds a new leaf entry to the DIT
Remove entry	Removes a leaf entry from the DIT
Modify entry	Modifies attributes of a particular entry
Modify RDN	Modifies the last RDN of a leaf entry

1.4 Standardized Features of the Directory Service

Directory service users are concerned with goal-oriented tasks (such as looking up a name or a telephone number). The directory service provides the following features (defined in the X.500 standard):

- User-friendly naming
- Lookup
- Searching
- Browsing
- Groups
- User identification (authentication)
- Routing of requests

1.4.1 User-Friendly Naming

Objects can be referred to with names that are readable. These names are assigned to ensure, as far as possible, that they can be predicted or remembered by human users. When sending a message by means of electronic mail, it is ill-advised to specify a confusing combination of figures, letters, and special characters as the receiver because they are open to misinterpretation. This information, which is necessary for sending but is of no interest to the user, is concealed in the directory service. User-friendly names typically consist of attributes that are inherent to the object, and not fabricated to suit the method of transmission.

This user-friendly naming is enhanced by the use of the alias names. One administrator can use the names "laser printer" and "line printer" whereas another can call them "high- quality printer" and "high-speed printer."

1.4.2 Lookup

Users can select the DN or alias name of an object together with the types of attributes required. The directory service returns all values associated with these attribute types. This function is similar to the white pages of the telephone directory. However, it supplies considerably more information than just addresses and telephone numbers. For example, an entry for a program name not only indicates what a program can do, but also where it is installed and who has access rights.

1.4.3 Searching

It is possible to search for objects according to common characteristics in a fashion similar to using the yellow pages in a telephone directory. As a general rule, the search is carried out from a particular entry onward and, in extreme cases, from the root onward.

Filters are used to specify search criteria; that is, only entries that satisfy the filter are returned. Both the search time and the number of hits in a successful search operation can be specified by the user. The user could use this function of the directory service to obtain, for example, a list of all laser printers installed in one particular building.

1.4.4 Browsing

A user who does not know the DN or the alias name of the object to be located can find the necessary information by ''browsing'' through the DIT. The user selects a specific entry, and all entries located immediately below it in the hierarchical structure are then displayed. The user then selects a new entry from these entries (this is known as *recursive listing*).

1.4.5 Groups

A *group* is an object consisting of a collection of object names. Object names within groups can also form groups. For example, groups can be used to make distribution lists.

1.4.6 User Identification (Authentication)

The directory service supports user authentication in a number of ways (for example, simple authentication with name and password and strong authentication using public key cryptography).

1.4.7 Routing of Requests

If a DSA cannot find the requested information, the following mechanisms are available to handle the request:

Referral The DSA sends the requesting DUA or DSA a reference indicating a DSA to be addressed as an alternative. The referral is handled using the DAP or DSP.

Chaining The DSA passes the request directly on to another DSA using the DSP.

Multicasting The DSA passes the request directly on to several other DSAs using the DSP.

1.5 Extensions to the X.500 Directory Service

In addition to the functions standardized by X.500, the following features are also available in GDS:

- Shadow Information
- Modeling of Knowledge Information
- Access Control
- Directory User Agent Cache
- Remote Administration
- Schema Information
- Tree Processing

1.5.1 Shadow Information

Frequently used data that is stored on one DSA can be copied and updated by other DSAs. This functionality increases the availability of the data and reduces the network access and costs. A temporary inconsistency in the data, which can occur, is tolerated by the directory system. If this inconsistency is unacceptable, only master information can be used. The **Master Knowledge** attribute, which is part of every object, indicates whether the data is master or shadow information.

Shadow administration involves the creation and manipulation of shadows and shadowing jobs. A shadowing job is stored in a local shadowing job file and contains the following information:

- Name of an object or subtree that has been replicated on a specific target DSA
- Name of the target DSA that has the replica
- Update frequency for the object or subtree on the target DSA

In the GDS implementation, a daemon process periodically updates an object or subtree on a target DSA by looking at the information stored in the shadowing job file. Note that a shadowing job should not be confused with a process or system task. A shadow is a read-only (in most cases) replica of

some object or subtree maintained in a nonlocal DSA that is updated at specified intervals.

The administrator has the option of activating a shadowing job (if one does not already exist), setting the update frequency of an existing job, or deactivating a job. The administrator selects the update frequency from one of three values: HIGH, MEDIUM, or LOW. A series of exact time values, which can be specified, is associated with each one of these values:

- HIGH — 5 to 30 minutes
- MEDIUM — 1 to 12 hours
- LOW — daily, weekly, twice a week

An administrator should choose the high update frequency if several objects change over a very short time in the DSA and the entries need to be updated as soon as possible. A low update frequency is used when there are only a few changes over the DSA in a short period of time.

An administrator can perform shadow administration while connected to the local DSA containing the objects to be administered. Shadow administration allows the administrator to create and delete shadows of objects or subtrees in a DSA, to create and delete shadowing jobs, and to manage shadowing jobs.

1.5.2 Modeling of Knowledge Information

GDS models the knowledge information so that each DSA contains an entry with its own name (for example, **/C=DE/O=Smith Ltd/OU=DI/CN=DSA/CN=DSA1**) and its Presentation Service Access Point (PSAP) address (which is its network address) as attributes.

The **Master Knowledge** attribute contains the DN of the DSA that masters the object. This is shown in Figure 1-11 with the names of the DSAs and the objects they master:

- **/C=DE/O=Smith Ltd/OU=AP2** is mastered by **DSA2.**
- **/C=DE** is mastered by **DSA1.**
- **/C=DE/O=Smith Ltd** is mastered by **DSA1.**
- **/C=DE/O=Smith Ltd/OU=AP2/CN=M** is mastered by **DSA1.**

Each DSA extracts its own name from a configuration file. By comparing this name with the **Master Knowledge** attribute of an entry in the DSA, the DSA can determine whether it masters the entry or just a shadow copy of the entry.

When a master entry is created on one DSA and the entry's superior node is mastered by another DSA, a *mandatory shadow* of that entry is created automatically on the DSA that masters the superior node.

For example, as shown in Figure 1-11, the entry for **/C=DE/O=Smith Ltd/OU=AP2** is mastered by **DSA2**. When the administrator adds the master entry to **DSA2**, a mandatory shadow of **/C=DE/O=Smith Ltd/OU=AP2** is created automatically on **DSA1** (as shown by the triangle indicating a mandatory shadow entry). Note that the three subordinate master entries under **/C=DE/O=Smith Ltd/OU=AP2** in the tree do not appear as shadow entries on **DSA1**. That is because their superior node, **/C=DE/O=Smith Ltd/OU=AP2**, is mastered by the same DSA.

Figure 1-11 also illustrates how a mandatory shadow is created on **DSA2**. The entry **/C=DE/O=Smith Ltd/OU=AP2/CN=M** is added to the tree by the administrator of **DSA1** beneath its superior node (and shadow entry) **/C=DE/O=Smith Ltd/OU=AP2**. A mandatory shadow of **/C=DE/O=Smith Ltd/OU=AP2/CN=M** is created automatically on **DSA2** under the master entry **/C=DE/O=Smith Ltd/OU=AP2**. The mandatory shadow is created because the superior node of the new master entry is mastered on another DSA (**DSA2**).

The reverse works with respect to removing entries from the directory. When a master entry is removed, its mandatory shadow entry is removed too.

If a master entry contains a mandatory shadow when an entry is being modified, both entries are modified.

When the DIT is distributed over several DSAs, each DSA must contain the following:

- All the entries that it masters.

  This is shown for **DSA1** in Figure 1-11 by the circles denoting master information for the following objects:

  — **/C=DE**

  — **/C=DE/O=Smith Ltd**

— **/C=DE/O=Smith Ltd/OU=AP1**

— **/C=DE/O=Smith Ltd/OU=AP2/CN=M**

— **/C=DE/O=Smith Ltd/OU=AP1/CN=Huber**

— **/C=DE/O=Smith Ltd/OU=AP1/CN=Meter**

— **/C=DE/O=Smith Ltd/OU=AP1/CN=Laser Printer**

- If applicable, all shadow entries necessary to link the master entries to the root.

  This is shown in Figure 1-11 by the inclusion of the shadow entry **/C=DE/O=Smith Ltd/OU=AP2** (with the name of the DSA that masters it, **DSA2**, in parentheses) to specify the master entry **/C=DE/O=Smith Ltd/OU=AP2/CN=M**.

- All direct children of master entries.

  This is shown in Figure 1-13 in the box containing **DSA1**. **/C=DE/O=Smith Ltd/OU=AP1** and **/C=DE/O=Smith Ltd/OU=AP2** are direct children of **Smith Ltd**.

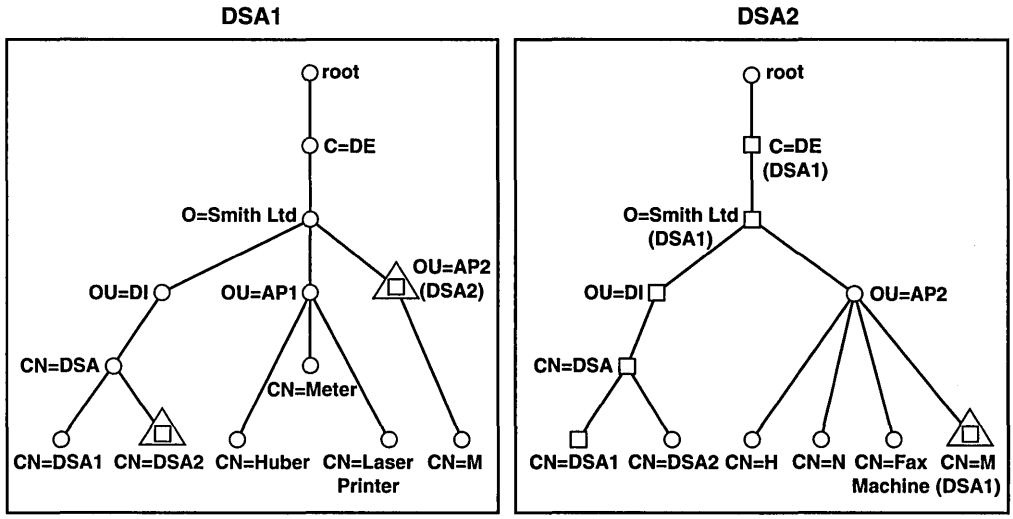
- The shadows of the DSA entries that appear as **Master Knowledge** attributes in the shadows available in this DSA.

  For example, in order to return referrals or to perform chaining, the DSA must know the appropriate DSAs. The DSA's name must be kept as master, and all the other DSAs are stored in this DSA as shadows.

  In Figure 1-11, **DSA1** retains its name as master and the name of **DSA2** as shadow. **DSA2** holds its name also as master and the name of **DSA1** as shadow. If an administrator looks for children of **/C=DE/O=Smith Ltd/OU=AP2** (on **DSA1**) and requests master information, **DSA1** returns **/C=DE/O=Smith Ltd/OU=AP2/CN=M**. **DSA1** uses the master knowledge of **/C=DE/O=Smith Ltd/OU=AP2** to determine which DSA needs to be contacted. To contact this DSA, it needs to know that DSA's PSAP address (which is guaranteed by the existence of a shadow entry of **DSA2**). This enables a DSA to take part in navigation through chaining and referrals.

  (The DSA can also hold additional shadows for data replication purposes; see Section 1.5.1.)

Figure 1–11. Storage of Knowledge Information in GDS



Legend:

○ = Master Entry.

□ = Copy with Master Knowledge information.

△ = Mandatory Shadow (automatically generated).

1.5.3 Access Control

In addition to authentication (by means of name and password), access protection is required for each object at attribute level. A telephone number, for example, is an attribute that, in general, everybody is allowed to read. However, attribute values such as number of children, bank accounts, or salary groups are restricted to a limited number of people. In addition, even for attributes that everyone is allowed to read, it is only acceptable for a few people to have authorization to change the values.

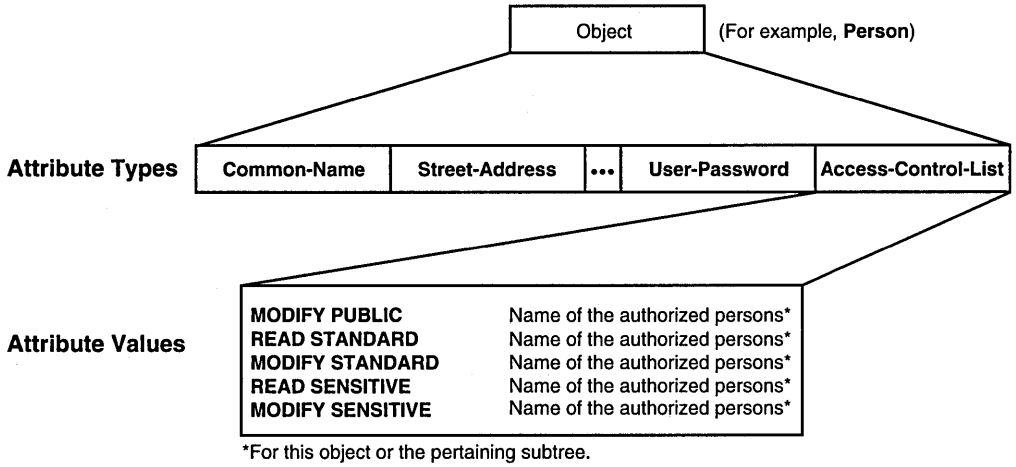
Because there can be many different attributes in the DIT, it is too expensive to define a protection mechanism for each individual attribute type. Instead, the attributes are divided into three classes:

- Public
- Standard
- Sensitive

Read and modify access rights can be defined for each of the three classes. The values for the **Access-Control-List** (**ACL**) attribute for an entry specify which person has which type of access to the individual attributes. Each entry has an **ACL** attribute that has multiple values. Each value has a list of names associated with one of the three access classes **PUBLIC**, **STANDARD**, and **SENSITIVE**, and **READ** and **MODIFY** access rights; this provides a total of five possible lists. (**READ PUBLIC** is not necessary.) These lists are used to control access to other attributes in the entry. For example, to modify the public attribute **Telephone-Number** you must be on the **MODIFY PUBLIC** list in the ACL of the specific entry.

Figure 1-12 shows the values of the ACL attribute. (The class of authorization **READ PUBLIC** does not need to be entered because, by definition, **PUBLIC** attributes can be read by all users.)

Figure 1–12. Access Control Using the ACL Attribute



The following specifications apply for assigning access rights:

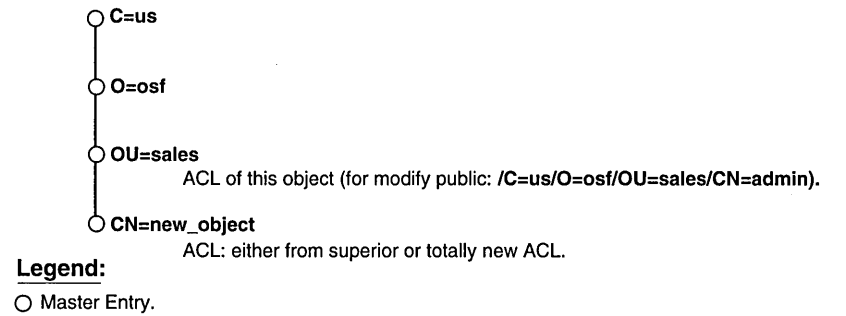
- The ACL of the default schema has no access rights when GDS is configured. Every user, including the anonymous user (one without a DN and password), has read and write access to all attributes in the schema.

  The administrator needs to remember that any anonymous user who has access to the GDS administration programs can log into a remote DSA. However, only **PUBLIC** attributes that are not protected by an ACL can be read and modified. It is strongly recommended that administrators protect objects by entering their DNs in the ACLs of objects requiring some form of restricted access. Otherwise, an unauthorized user could change or destroy an object not protected in this manner.

- A master entry can only be created by the user who has write access to the naming attribute of the parent node. The user creates all attributes of the entry and establishes which objects can access these attributes in the **ACL** attribute. If the user does not enter an **ACL** attribute when creating an entry, GDS automatically enters the **ACL** attribute of the parent node for the new entry.

  In Figure 1-13, **CN=new_object** can only be created if the administrator logs in as **/C=us/O=osf/OU=sales/CN=admin**. If no ACL is specified, the new object is assigned the **ACL** attribute of **/C=us/O=osf/OU=sales**; otherwise, it takes the ACL that the administrator specified for the operation.

Figure 1–13. Assigning ACLs for an Add Operation



- A master entry can only be deleted by users who have modify access to the naming attribute of the entry to be deleted.

In Figure 1-13, **CN=new_object** can only be deleted if the administrator logs in as **/C=us/O=osf/OU=sales/CN=admin**. The object can only be modified if the administrator logs in as **/C=us/O=osf/OU=sales/CN=admin**.

- A shadow entry created by means of shadow handling (see Chapter 10) has the same **ACL** attribute as the corresponding master entry. Therefore, such an entry can only be modified and deleted by users who are also permitted to modify and delete the master entry.

1.5.4 The Directory User Agent Cache

The Directory User Agent (DUA) cache is a process that stores a cache of information obtained from DSAs. One DUA cache runs on each client machine and is used by all the users on that machine. The DUA cache contains copies of recently accessed object entries and information about DSAs. The user specifies which information is to be cached. It is also possible to bypass the DUA cache by obtaining information directly from a DSA. This is desirable, for example, if the user wants to ensure that the information obtained is up-to-date.

A cache process also runs on a client/server machine. It contains the client address, the name of the local DSA, and the DSA's PSAP address (PSAP addresses are described in Chapter 2), which is required to get GDS running.

Unlike the DSA, the cache knows nothing about a schema or tree structure. Therefore, information without existing superiors can be added to the cache, and objects can be deleted while subordinates still exist.

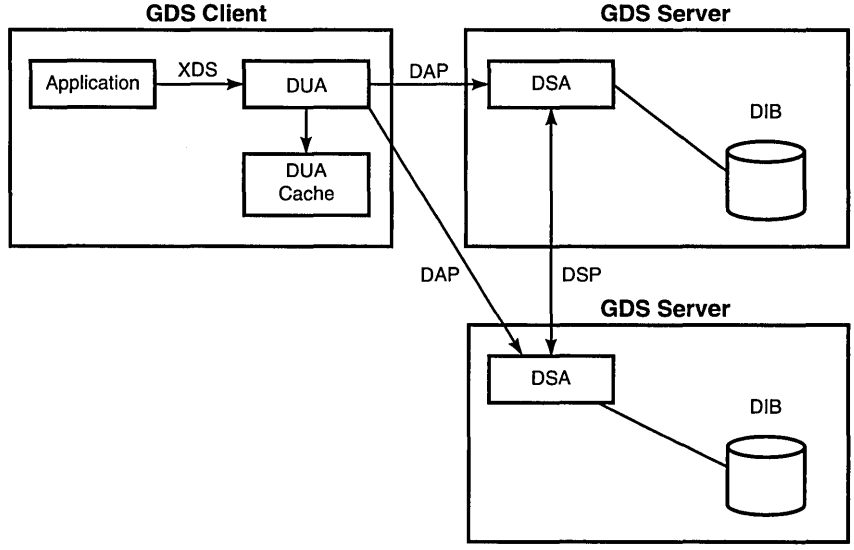
The cache does not handle ACL attributes. If an application reads data from a DSA and stores it automatically in the cache, only attributes that are classified as **Public** are stored in the cache.

The **Cache Update** process updates replicated information in a DUA cache. It runs as required and then terminates. The **Cache Update** process runs on GDS client and client/server machines.

An administrator (or any user who has access to the GDS administration tools) controls the information stored in the cache by logging into the local machine using the option in Mask 1 **Logon to the DUA cache**. (Masks are menus that enable a user to enter input interactively in the GDS administration programs. Masks are described in detail in Chapters 8 through 11.)

Figure 1-14 shows the interaction between an application program, using the XDS interface, and the GDS client and server. The GDS client and server use the DAP to communicate. The GDS servers use the DSP to communicate with one another. The DAP and DSP perform functions similar to the function that DCE RPC protocols perform in other DCE services.

Figure 1–14. GDS Components



1.5.5 Remote Administration

Only local administration is defined as standard. This means that each DSA must be administered separately. For maximum flexibility, GDS provides remote administration. This enables administrators to log into other DSAs and perform directory operations from one workstation. GDS enables remote administration of objects, schema, copies, and subtrees.

The administrator remotely logs into a DSA through the User Identification mask (Mask 1). The administrator enters a DN that represents the administrator and a password (if required). The DN must exist in the directory tree and have modify access to any object (as defined in the ACL of that object) the administrator wants to perform object, schema, subtree, or

shadow administration functions on. (However, modify access is only required if the administrator wants to change anything in the DIT.) The next mask that is displayed, the Logon to a Specific DSA mask (Mask 2), enables administrators to enter the name of a remote DSA to which they want to be connected.

1.5.6 Schema Information

The *directory schema* governs the structure of the directory tree. It consists of a set of rules that defines the name structure, the object classes, and the attribute types and their syntaxes. The directory standard only describes the schema concept and not the schema structure. It does, however, make recommendations on the attributes types and object classes that need to be present in the directory.

In GDS, the schema is stored as an object in the directory directly under the root with the following **distinguished name : /CN=Schema**. A default schema is supplied to initialize a directory service system.

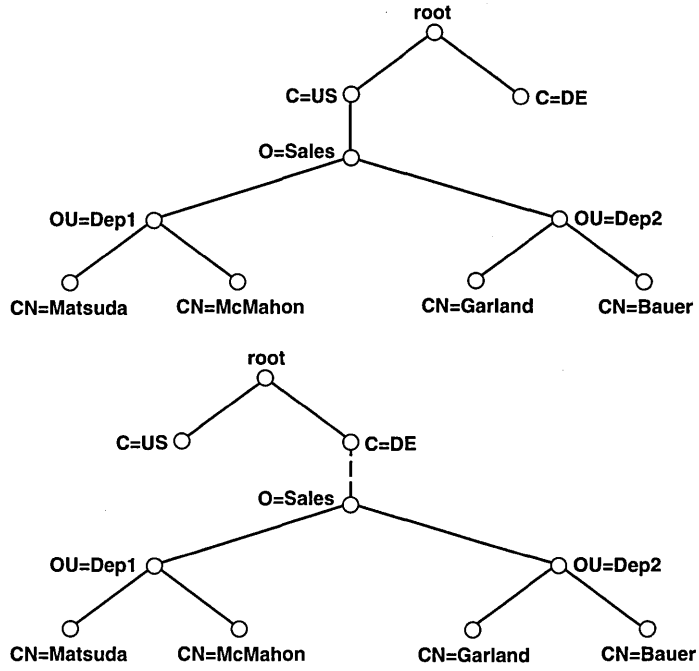
1.5.7 Tree Processing

The current standard defines only the insertion, deletion, and renaming of end nodes (leaves). In addition, GDS administration includes a series of tree processing functions that can be used to restructure the DIT.

These tree processing functions allow the user to save, append, move, copy, and delete subtrees, to change the **Master Knowledge** attribute of all objects in a subtree, and also to modify a special value of an existing attribute in a subtree.

Figure 1-15 illustrates how tree processing functions could be used to to delete a subtree from one node and append it to another.

Figure 1–15. Moving a Subtree



1.6 Schema

The structure of directory information is governed by a set of rules called a *schema*. A schema specifies rules for the following:

- The structure of the DIT
- The contents of entries in terms of attributes
- The syntax of attribute values and rules for comparing and matching them

1.6.1 The GDS Standard Schema

DCE includes a default or "standard" schema for GDS. This is the GDS proprietary interpretation of the X.500 schema.

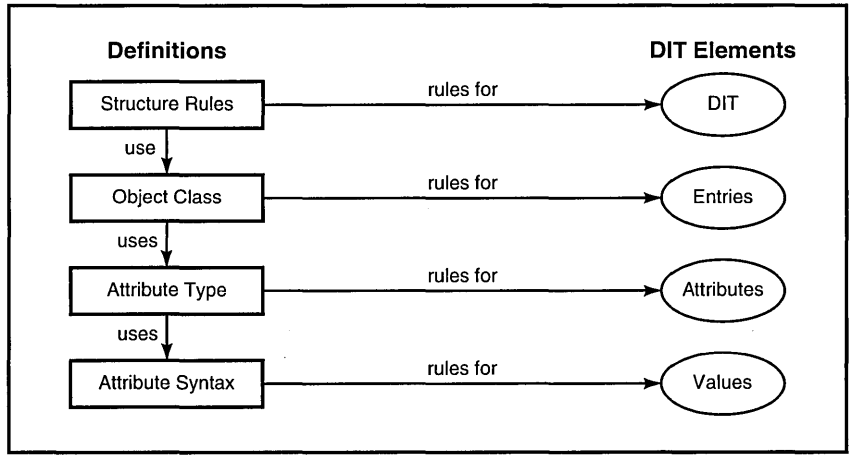
Each attribute in the schema is assigned a unique object identifier and the syntax of its value. In addition, the schema specifies the mechanism for comparing attributes of this type with one another. Each entry in the DIT belongs to an object class governed by the schema. Object class definitions can be used to derive subclasses, supporting the inheritance and refinement of the attribute types defined for the superclass.

The ability to define subclasses is a powerful feature of the directory service. Structure rules determine which object classes are children and which object classes are parents in the DIT. Therefore, they define possible name forms.

The directory standard defines a number of standard attribute types and object classes. For example, it defines the attribute types **Common-Name** and **Description**, and the object classes **Country** and **Organizational-Person**. Based on the standard rules, GDS applications may enhance the schema with additional attribute types and object classes.

Figure 1-16 shows the relationship between schemas and the directory information model.

Figure 1–16. The Relationship Between Schemas and the DIT



The GDS standard schema includes the following tables which define the structure of the DIT:

- Structure Rule Table (SRT)
- Object Class Table (OCT)
- Attribute Table (AT)

1.6.2 The Structure Rule Table

The Structure Rule Table (SRT) defines the hierarchical relationships that are permitted between objects and their RDNs. The SRT supplied with the GDS standard schema contains the entries shown in Table 1-2.

Table 1–2. SRT Entries (for DSAs)

Rule Number	Superior Rule Number	Acronym of Naming Attribute	Acronym of Structural Object Class
1	0	CN	SCH
2	0	C	C
3	2	O	ORG
4	3	OU	OU
5	4	CN	ORP
6	4	CN	ORR
7	4	CN	APP
8	4	CN	MDL
9	4	CN,OU	APP ORP
10	7	CN	APE
11	7	CN	DSA
12	7	CN	MMS
13	7	CN	MTA
14	7	CN	MUA
15	2	L	LOC
16	15	CN	REP
17	15	CN,STA	REP

The SRT determines how the object classes are laid out in the DIT by assigning rule numbers to each object class. An object class's Superior Rule Number specifies the object class directly above it in the DIT.

For example, the object class **Organization** (abbreviated with the acronym **ORG** in the SRT), has a Superior Rule Number of 2, indicating that it is located in the DIT beneath the object class **Country** (**C**) which has a Rule Number of 2. **Organization Unit** (**OU**) is located beneath **Organization** because it has a Superior Rule Number of 3, and so on.

The SRT only contains structured object classes (that is, classes forming branches in the DIT). Other object classes (such as abstract and alias classes) are not included.

The SRT specifies the attribute or attributes used to name entries belonging to each object class. These attributes, called *naming attributes*, are used to define the RDN and therefore the DN of directory entries.

There are actually two SRTs defined for GDS, one for DSAs and a slightly different one for the administration programs. The SRT for the GDS

administration programs is a compressed version of the DSA version and is updated to reflect any changes in the DSA schema. Instead of having 17 Structure Rules, one for each naming attribute, it has only 10. Where naming attributes share the same Superior Rule Number, the naming attributes are given the same rule number.

Figure 1-17 shows the structure of the Directory Information Tree as defined by the Structure Rule Tree of the GDS standard schema for a DSA. It corresponds to the SRT entries in Table 1-2. Note that in Figure 1-17, the SRT contains two entries for **Organizational-Person** that specify different sets of permitted naming attributes.

The SRT for the standard schema used by the GDS administration programs is shown in Table 1-3.

Figure 1–17. Structure of the DIT for GDS Administration Programs

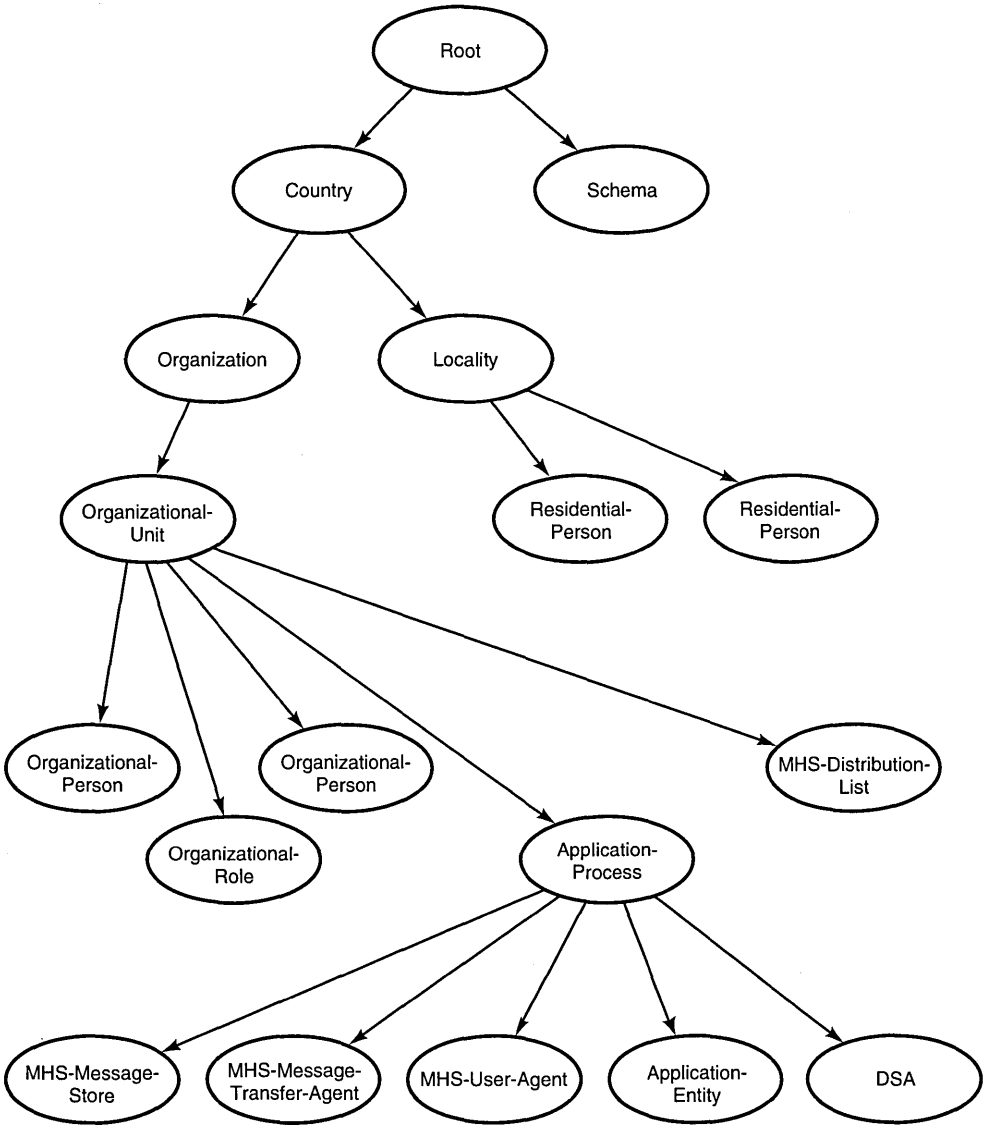
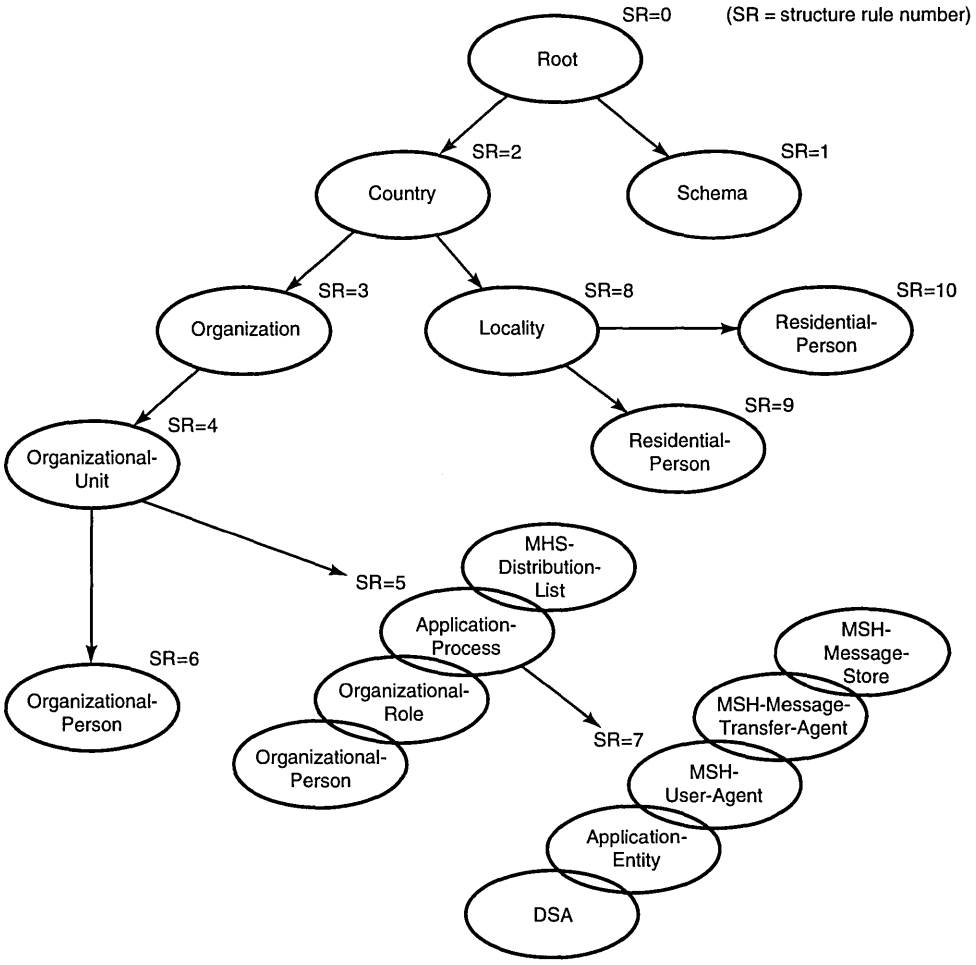


Table 1–3. SRT Entries (for GDS Administration Programs)

Rule Number	Superior Rule Number	Acronym of Naming Attribute	Acronym of Structural Object Class
1	0	CN	SCH
2	0	C	C
3	2	O	ORG
4	3	OU	OU
5	4	CN	ORP, APP, ORR, MDL
6	4	CN, OU	ORP
7	5	CN	DSA, APE, MMS, MTA, MUA
8	2	L	LOC
9	15	CN	REP
10	15	CN,STA	REP

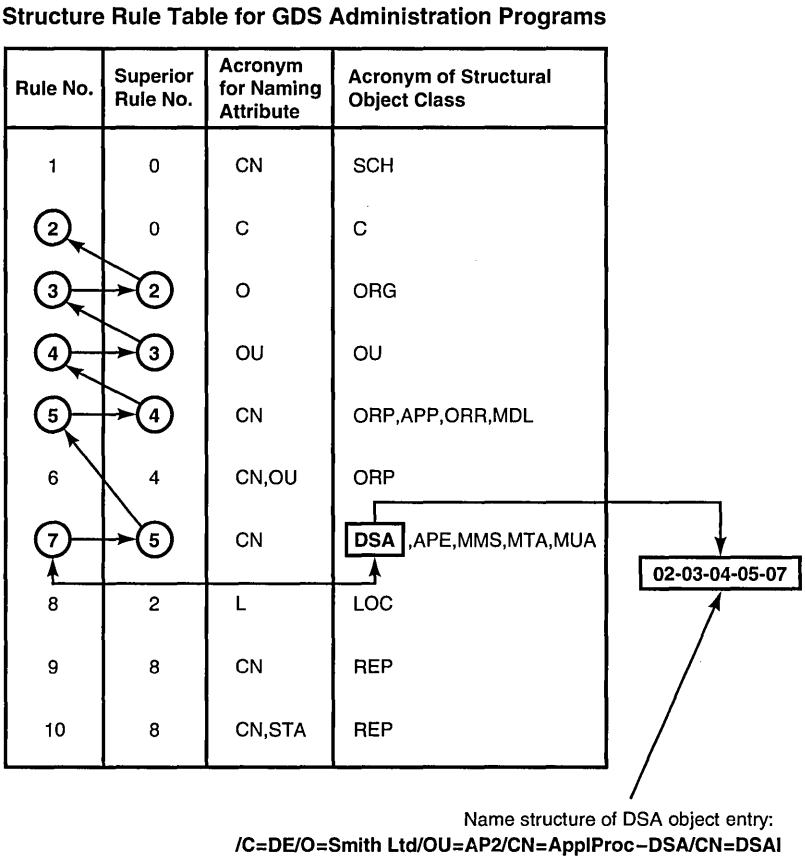
Figure 1-18 shows the structure of the DIT as defined by the SRT of the GDS standard schema for the GDS administration programs. It corresponds to the SRT entries in Table 1-3. Note that in Figure 1-18, the SRT contains two entries for **Organizational-Person** and **Residential-Person** that specify different sets of permitted naming attributes.

Figure 1–18. Structure of the DIT for Administration Programs



An administrator uses the structure rules to determine the name structure of an object entry so that the DN of the entry can be entered in administration program masks. For example, in Figure 1-19, the name structure of a DSA object entry **/C=DE/O=Smith Ltd/OU=AP2/CN=ApplProc-DSA/CN=DSA1** is 02-03-04-05-07. Starting from Structure Rule 7 (the Structure Rule for DSA), it is easier to work backward to successive superior rules to determine the correct name structure. Chapter 8 describes how an administrator uses these rules in the administration program to enter object entries into the directory.

Figure 1–19. Determining the Name Structure of a DSA Object Entry



1.6.3 The Object Class Table

The object classes that make up the GDS standard schema are defined in the Object Class Table (OCT), which describes each object class, including its mandatory and optional attributes and the inheritance of the attributes from other object classes. Table 1-4 contains a partial listing of the OCT. (See Appendix B for a complete listing of the OCT for the GDS standard schema.) Each column contains information about an object class entry in the schema.

Table 1–4. OCT Entries

Acronym of Object Class	Acronym of Super Classes	Object ID	Name of Object Class	Kind of Object Class	File No.	Acronym of Auxiliary Object Classes	Acronym of Mandatory Attribute	Acronym of Optional Attribute
TOP		85.6.0	Top	Abstract	-1	-	OCL	
GTP	TOP		GDS-Top	Abstract	-1	-		ACL MK
ALI	TOP	85.6.1	Alias	Alias	-1	-	AQN	
C	GTP	85.6.2	Country	Structural	1	-	C	DSC SG CDC CDR
LOC	GTP	85.6.3	Locality	Structural	4	-		DSC L SPN STA SEA SG CDC CDR
ORG	GTP	85.6.4	Organization	Structural	1	MUS	O	DSC L SPN STA PDO PA PC POB FTN IIN TN TTI TXN X1A PDM DI RA SEA UP BC SG CDC CDR
MUS	TOP	86.5.1.3	MHS-User	Auxiliary	-1	-	MOA	MDL MDT MDE MMS MPD

Note: All the object identifiers in Table 1-4 stem from the root **{joint-iso-ccitt(2) ds(5) objectClass(6)}**.

Column 2, Acronyms of Super Classes, provides the class from which an object class inherits its attributes. Using the information in Column 2, it is

possible to derive a graphical representation of the inheritance properties of object classes in the DIT, as shown in Figure 1-17.

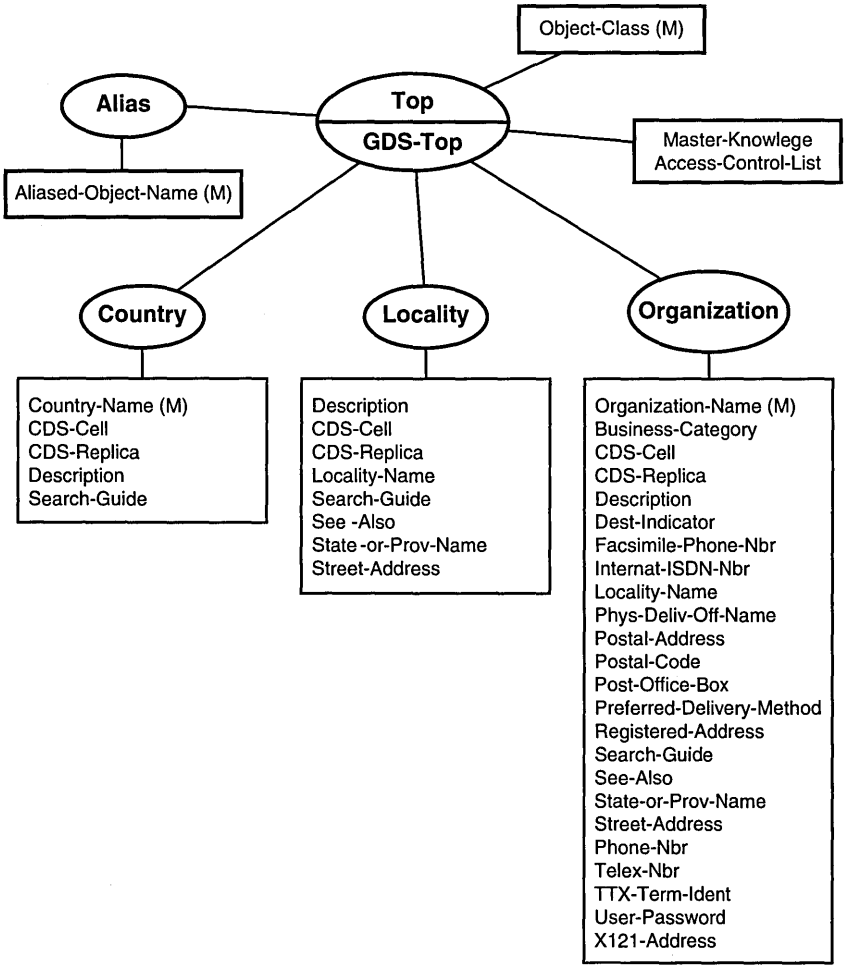
The object class, **Top**, is the root of the tree with **Alias** and **GDS-Top** as the main branches. **Top** contains the attribute type **Object-Class**, which is inherited by all the other object classes.

Do not confuse the information in the OCT with that presented in the SRT. There is no direct relationship between the relative location of branches and leaves in the DIT structure and the inheritance properties of classes with their superclasses and subclasses.

For example, when a directory service request such as a **search** operation is made by a directory user, the SRT is used by the directory service to indicate its position in the DIT. The directory service uses the information defined in the SRT to traverse the tree so that the requested object can be located in the directory. Figure 1-17 shows the object class **Organization** located beneath **Country** in the DIT.

On the other hand, the OCT defines, among other things, the attributes of an object class along with its inherited attributes from its superclasses. The superclasses, in turn, inherit the attributes from their superclasses, and so on until the root **Top** is reached (from which all classes derive their attributes). Figure 1-20 shows the object class **Organization** as a subclass of **GDS-Top**. As such it inherits its attributes from **GDS-TOP**, which in turn inherits them from its superclass, **Top**. **GDS-TOP** is an *unregistered object class* that is proprietary to GDS. As such it does not have an object ID and does not appear in any of the administration masks. The "M" in parentheses indicates a mandatory attribute.

Figure 1–20. Partial Representation of the Object Class Table



The OCT also contains the unique object ID (OID) of each class in the DIT except **GDS-TOP**, which has none. OIDs are described in Section 1.1.4. Table 1-5 shows some examples of OIDs for directory classes as defined in the X.500 standard.

Table 1–5. Object Identifiers for Selected Directory Classes

Object Class Type	Object Identifier
Alias	85.6.1
Application-Entity	85.6.12
Application-Process	85.6.11
Country	85.6.2
Device	85.6.14
DSA	85.6.13
Group-of-Names	85.6.9
Locality	85.6.3
Organization	85.6.4
Organizational-Person	85.6.7
Organizational-Role	85.6.8
Organizational-Unit	85.6.5
Person	85.6.6
Residential-Person	85.6.10
Top	85.6.0

Note: All the object identifiers in Table 1-5 stem from the root **{joint-iso-ccitt(2) ds(5) objectClass(6)}**. To enable more efficient transmission over a communications line, the first two subidentifiers, **2** and **5**, are coded together which results in one subidentifier, **85**.

1.6.3.1 Object Class Types

Object classes in the OCT are divided into four categories:

- An *abstract* object class provides basic relationships on subclass object classes, but has no entries itself. Typical examples of these subclasses are **Top**, **GDS-Top**, or **Person**.
- An *alias* object class indicates that an entry is an alias entry. In the GDS implementation, there is only one alias object class, which is the object class **Alias**.
- A *structural* object class is defined to be used in the structural specification of the DIT.

Only a structural object class can be referenced in the SRT. For example, **Organization** is a structural object class, while its superclasses **GDS-Top** and **Top** are not. The third entry in Table 1-3 references **Organization** as a structural object class by the acronym **ORG**; however, the object class **Top** with the acronym **TOP** is not referenced in any entry of the table.

- An *auxiliary* object class provides attributes that are not included in a structural object class.

An object entry belongs to exactly one structural object class. For example, an object, whose distinguished name is ruled by the fifth entry of the SRT in Table 1-3 can belong to only one of the structural object classes **Organizational-Person (ORP)**, **Application Process (APP)**, **Organizational Role (ORR)** or **MHS Distribution List (MDL)**.

An object that belongs to an object class also belongs to all its superclasses. Therefore, an object of object class **Organization** belongs at least to the abstract object classes **Organization**, **GDS-Top** and **Top**. This is reflected by the values of its object class attribute, which are the object identifiers **85.6.4** for **Organization** and **85.6.0** for **Top**. **GDS-Top** as a proprietary object class has no object identifier and is not among the values of the object class attribute.

As described in Section 1.6.3.2, object classes list the attributes that may be assigned to objects. Some applications of the directory may need to specify object attributes for objects that are not listed in the structural object class or its superclasses. These applications can define auxiliary object classes, which list these attributes.

For example, message handling systems use the auxiliary class **MHS-User** to specify a package of message handling attributes for objects of different object classes. In Table 1-4, **MHS-User** is assigned to the structural object class **Organization** as an auxiliary object class. It could be assigned to other structural object classes (such as **Locality**) also.

Therefore, in addition to being a member of a structural object class, an entry can be a member of one or more auxiliary object classes. The acronyms of the auxiliary object classes permitted for the entries of a particular structural object class are listed in the OCT entry of the structural object class. Hence in our example an **Organization** may be an **MHS User** or not, as indicated by its object class attribute. If the object class attribute

has a value **86.5.1.3** in addition to the values mentioned above, the organization is also an **MHS User** and the message handling attributes may be assigned to it.

The file numbers in the OCT entries tell the DSA which objects it has to store in the same C-ISAM files. Table 1-4 shows that **Countries** and **Organizations** are stored in the same file, while **Localities** are stored in a different file. The file number is ignored for object classes that are not structural. When object classes share most of their attributes, they should share the file number to increase performance. When many of the attributes differ in the mandatory and optional attribute sets (described in Section 1.6.3.2), the object classes should not share the file number in order to save space on disk.

1.6.3.2 Mandatory and Optional Attributes

Another important feature of the OCT is the distinction made between mandatory and optional attributes for each object class. This distinction is based on definitions in the X.500 standards documents. These documents (Recommendations X.520 and X.521) recommend selected object classes and associated attribute types that use ASN.1 notation. Each object class has one or more mandatory attributes associated with it, to be used by the implementors who want to comply with the X.500 standards recommendations. Optional attributes are also defined.

For example, the class **Country** must contain the mandatory attribute **Country Name** (or **Country-Name** as defined in the GDS standard schema), and can contain the optional attributes **Description** and **Search-Guide**. The DCE implementation also adds two more attributes, **CDS-Cell** and **CDS-Replica**, to incorporate other aspects of the DCE environment that are implementation-specific.

Country is assigned the OID 2.5.6.2 (85.6.2). This number distinguishes it from the other object classes defined by the standard. The **Top** superclass is designated as 2.5.6.0 (85.6.0). The first three numbers, 2.5.6 (85.6), identify the object class as a member of a discrete set of object classes defined by X.500. The last number in the OID distinguishes objects within that discrete set. **Alias**, a subclass of **Top**, is assigned the number 2.5.6.1 (85.6.1). **Country** is assigned the number 2.5.6.2 (85.6.2), and so on. **GDS-Top** has no OID, because it is implementation-specific and, as such, is not identified by the standard.

1.6.4 The Attribute Table

The Attribute Table (AT) defines the attributes that constitute the entries in the GDS standard schema. (See the *OSF DCE Administration Guide — Core Components* for a complete listing of the AT.) The OIDs range from 85.4.0 to 85.4.35, as defined by the X.500 standard, 86.5.2.0 to 86.5.2.10, as defined by the X.400 standard, and include additional OIDs for GDS-specific attributes.

Table 1-6 shows a partial listing of the AT for the GDS Standard Schema.

Table 1–6. AT Entries

Acronym of Attribute	Object ID	Name of Attribute	Lower Bound	Upper Bound	Maximum Number of Values	Syntax	Phonetic Flag	Access Class	Index Level
OCL	85.4.0	Object-Class	1	28	0	2	0	0	0
AON	85.4.1	Aliased-Object-Name	1	1024	1	1	0	0	0
KNI	85.4.2	Knowledge-Information	1	1024	0	4	0	0	0
CN	85.4.3	Common-Name	1	64	2	4	1	0	1
SN	85.4.4	Surname	1	64	2	4	1	0	0
SER	85.4.5	Serial-Number	1	64	2	5	0	0	0
C	85.4.6	Country-Name	2	2	1	1010	1	0	1
L	85.4.7	Locality-Name	1	128	2	4	1	0	1
SPN	85.4.8	State-or-Province-Name	1	128	2	4	1	0	0

The **Lower Bound** and **Upper Bound** columns specify the maximum or minimum number of bytes (or octets) that the value of an attribute can contain. In the **Maximum Number of Values** column, the schema puts constraints on the number of values that an attribute can contain. A 0 (zero) in this column means that there is no restriction on the number of values.

The **Syntax** column describes how the data is represented and how it relates to **ASN.1** syntax definitions for attributes. The **Common-Name** attribute is defined as case-insensitive. The size of the string ranges from 1 to the upperbound value defined by the schema in the **Upper Bound** column for the **Common-Name** attribute (in this case, 64 bytes or octets).

Note: The **Common-Name** attribute is assigned the number 3 as standard. This corresponds to the 3 in the OID 85.4.3.

As mentioned previously for object classes, OID values specified in the AT are defined as constraints in the GDS header files.

An **Access** class is assigned for each attribute (**Public(0)**, **Standard(1)**, **Sensitive(2)**). An administrator can change this value for any attribute by using the schema administration operations described in Chapter 9.

The other columns in the AT refer to the maximum number of permitted values, phonetic matching, and index level. The information contained in these columns is described in Chapter 9.

1.6.5 Syntaxes

An attribute syntax defines the syntactic rules for the attribute values for each attribute type in the directory. This syntax includes the data type in **ASN.1** and, generally, one or more of the matching rules used to compare values.

As shown previously, the AT has a **Syntax** column. This column contains a number that corresponds to a particular syntax, which describes how that data value is represented for a specific attribute type. In Table 1-6, for example, the **Syntax** column gives the value 2 for the **Object-Class** attribute, indicating that the syntax type is Object Identifier. (See the *OSF DCE Application Development Guide* for more information about syntaxes).

(See Chapter 9 for a description of the syntaxes that can be applied.)

Chapter 2

GDS Components

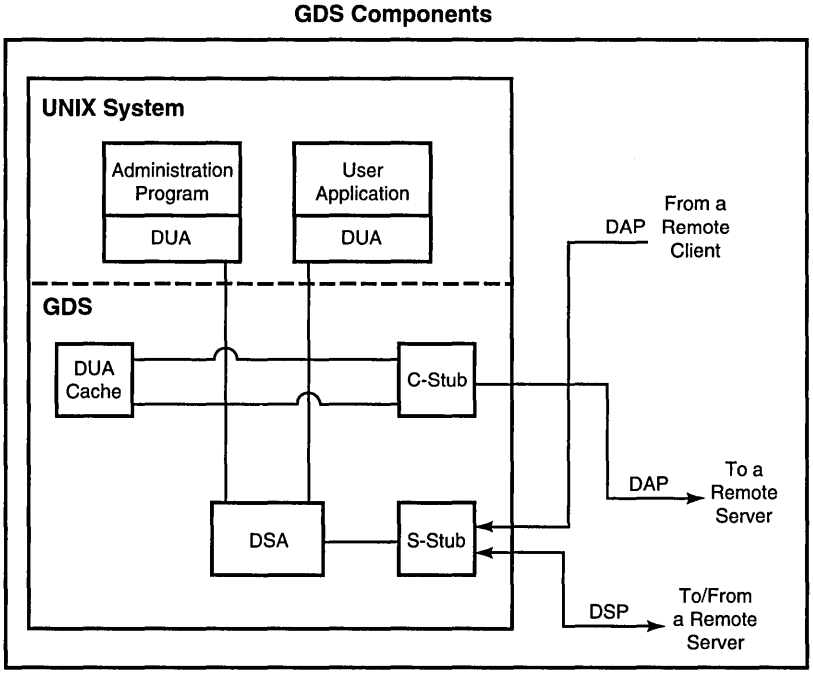
This chapter describes the components of GDS and how they work.

2.1 Client/Server Model

GDS is based on the client/server model. In this model, a distributed application (in this case GDS) is divided into two parts, one part residing on each of the two computers that communicate during the exchange of information. The client side of the application is the part that resides on the node that initiates the distributed request and receives the benefit of the service. The server side of the application is the part that resides on the node that receives and executes the distributed request.

Figure 2-1 shows how the client/server model is implemented for GDS.

Figure 2–1. GDS Components



The client consists of the following (UNIX) processes:

- An application that links the XDS library (the **gdsditadm** program is an example of such a process).
- The C-stub that handles the connection over the communication network for accessing a remote server. It implements the upper layers of the ISO protocol stack (described in Section 2.3.1). Its function is similar to the RPC Runtime (GDS uses OSI protocols instead of DCE RPC).
- The DUA cache.

The server consists of the following (UNIX) processes:

- A DSA that accesses the database.
- An S-stub that handles the connection over the communication network so it can either access a remote server or be accessed by a remote server

or client. The S-stub is similar to the C-stub, except that it runs on the server machine and manages its communications with DUAs and other DSAs.

Within the same system, the application (DUA) accesses the DSA directly by bypassing the C-stubs and S-stubs. The DUA cache is available to every client for fast access to frequently required information.

In addition, an interprocess communication (IPC) monitoring process monitors the interprocess communication between the various components.

All of these processes (apart from the application process) run continuously in the background as long as the directory system is active.

The following background processes are started when needed:

Shadow update process
To update the shadows of the directory in the DSAs, users can generate a shadowing job for an object or a subtree and for a target DSA. The system starts a corresponding shadow update process either periodically or on request and updates the shadows in the target DSA.

Cache update process
There is one local shadowing job for updating the entries in the DUA cache. The system starts a corresponding cache update process either periodically or on request and updates all entries in the DUA cache.

In addition to the background processes, the following administration programs are available:

gdssysadm Supports administration of the local GDS installation, such as configuration, server activation, and backup.

gdsditadm Supports administration of the contents of a GDS database and the local cache.

gdscacheadm
Supports administration of the local DUA cache (only necessary on client systems without **gdsditadm**).

2.2 XDS Application Program Interface

The X/Open Directory Service (XDS) is an application programming interface based on X/Open standards specifications. The XDS API consists of a library of functions for developing applications that access the directory service. GDS uses the XDS API internally to provide the functions available in the GDS administrative programs. Application programmers can use the XDS API to develop their own customized applications.

DCE programmers use the XDS API to make directory service calls. In DCE, XDS API directs the calls it receives to either GDS or CDS by examining the names of the information objects to be looked up. The XDS API contains functions for managing connections that use a directory server, namely, reading, comparing, adding, removing, modifying, listing, and searching directory entries. These functions map to the operations standardized by X.500 (as shown in Table 1-1).

The GDS Extension package and the MHS Directory User package provide additional information objects for use by security, cache management, and electronic mail applications when GDS is being used.

(See the *OSF DCE Application Development Guide* for more information on programming with XDS API.)

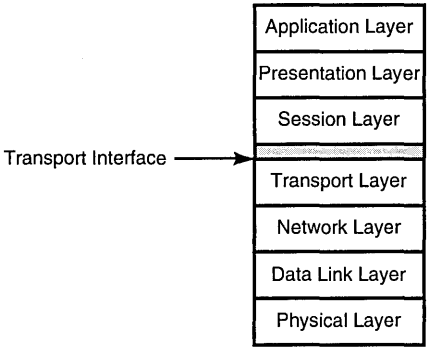
2.3 GDS Client/Server Communication

The X.500 directory service standard is written with a view to running on top of the Open Systems Interconnection (OSI) communications protocols. The OSI protocols are divided into seven layers:

- Physical
- Data Link
- Network
- Transport
- Session
- Presentation
- Application Layers

The upper three layers are implemented as libraries that are linked together with the C-stub and S-stub. The lower layers are part of the operating system and their services are made available to the upper layers through a transport interface. The transport interface is the shaded area in Figure 2-2.

Figure 2–2. The OSI Protocol Layers



2.3.1 Upper Layers

The directory service is an application layer protocol. Its specification requires the use of two other application layer service elements—the Association Control Service Element (ACSE) and the Remote Operation Service Element (ROSE)—and of the underlying layers. ROSE and ACSE of the application layer are implemented in GDS by the Remote Operation Service (ROS) library. The OSI Session Service (OSS), which is in the session layer, is implemented in GDS by the OSI Session Service library. The Presentation Service in the Presentation Layer is implemented by the ASN.1 library.

2.3.2 Lower Layers

DCE assumes that the system it runs on provides support for transport layer communications. The OSI protocols running above the transport layer were originally designed to run over OSI transport protocols. Many DCE systems run the Transmission Control Protocol (TCP/IP); therefore, GDS provides the capacity for running over the TCP/IP transport protocol as specified in RFC 1006. In GDS, the default is the TCP/IP transport protocol. The X/Open Transport Interface (XTI) can be used. The environment variable **TSITYPE** specifies which nondefault transport protocol will be used. If **TSITYPE** is **SOCKET**, TCP/IP is used. If **TSITYPE** is **XTI**, the XTI Transport Interface is used.

2.3.3 Client/Server Addresses (PSAPs)

OSI uses Service Access Points (SAPs) for addressing. A SAP is an abstract point at which a particular service is provided between two layers in the OSI protocol stack.

The administrator needs to know the PSAP address of each server machine in order to integrate it into the overall distributed directory system. A PSAP address is composed of one or more NSAP (Network Service Access Point) addresses, and the Presentation (P-Selector), Session (S-Selector), and Transport (T-Selector) selectors.

The administrator also needs to know the PSAP address of the client stub on the local machine so that it can be added to the DUA cache.The local DUA cannot access a remote DSA without knowing its client address.

NSAP addresses are intended to be globally unique. Each NSAP address identifies a particular computer system somewhere in the world. Various registration authorities are responsible for maintaining the uniqueness of these network addresses. An administrator must apply to these registration authorities to receive a globally unique NSAP address. (See Appendix D for more information on NSAP address authorities.)

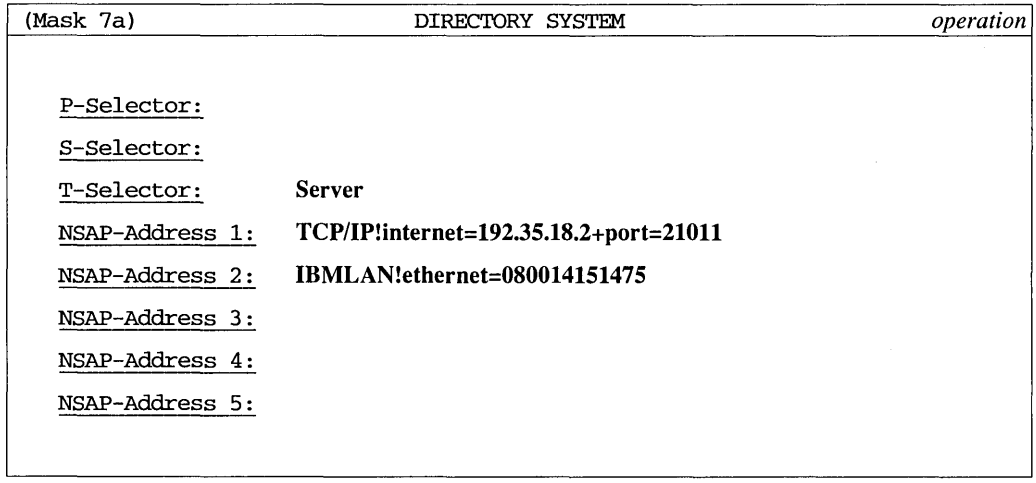
The P-Selectors, S-Selectors, and T-Selectors refer to the SAPs of upper layers in the OSI protocol stack within a given system. Unlike the NSAP address, these upper layer selectors need to be unique only within a particular system. They identify the application that communication is to be set up with.

GDS requires that the T-Selector be specified. Typically, the value of the T-Selector is **server** for a server and **client** for a client. However, an administrator can choose to configure more than one directory service, resulting in multiple directory IDs. Each directory service must have a unique T-Selector. For example, if two directory services are running on a specific system, two different PSAP address entries should be entered in the directory, in the DUA cache, or both, with the T-Selectors set to unique values. For example, one address can be assigned the T-Selector value **server1**, and the other a T-Selector value of **server2**.

The P-Selector and S-Selector fields are ignored by GDS. However, it is possible that a non-GDS service may interpret the P-Selector and S-Selector fields.

Figure 2-3 shows how a server address is entered in Mask 7a.

Figure 2–3. Example of a Server Address



The NSAP-addresses in Figure 2-3 show two different types of NSAP addresses: **NSAP-address 1** and **NSAP-address 2.**

NSAP-address 1 is a TCP/IP (socket) address. It is the responsibility of the administrator to ensure that the port numbers (entered in the figure as **port=21011**) are unique on their machines.

NSAP-address 2 is an example of a LAN address used on IBM RS6000 machines.

As mentioned previously, GDS operates over XTI as well as sockets. The environment variable **TSITYPE** determines which network is used. If the administrator wants to use both address types in an installation, both addresses can be stored in the DUA cache or DIT for the same object.

The structure and format of PSAP addresses is somewhat complex. Appendix D describes how these network addresses are derived and provides some basic guidelines on when and how to apply to registration authorities for a unique address.

2.4 Directory System Agents

DSAs are application processes that provides access to the DIB, to DUAs, and to other DSAs. A DSA may use information stored in its local database or interact with other DSAs to carry out requests. Alternatively, the DSA may direct a request to another DSA, which can help carry out the request.

This section describes three types of DSAs:

- Initial DSA
- First-level DSA
- Default DSA

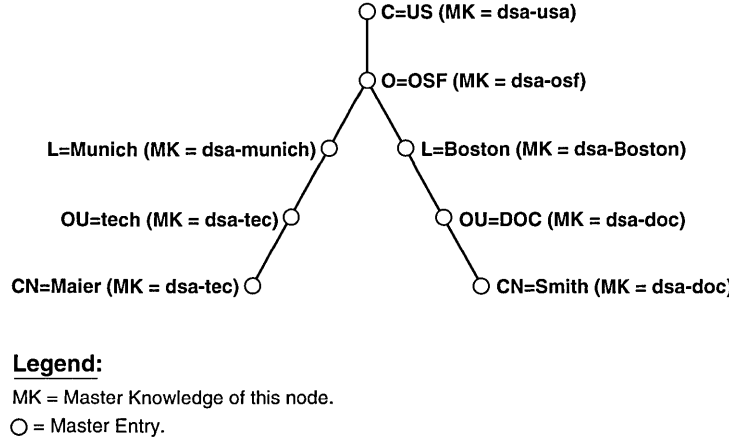
2.4.1 Initial DSA and Administrative Domain

Each DSA must contain the schema object under the root of the DIB. If a DSA has the master entry of the schema object, it is called an *initial DSA*. If other DSAs contain a shadow of the schema from an initial DSA, these DSAs constitute an administrative domain. To make a DSA part of an administrative domain, the administrator must perform special steps in the DUA cache and local DSA to initialize a client/server system.

For example, if the directory system uses the default schema, the administrator would enter the name of the initial DSA (including its PSAP address) in the DUA cache. The administrator would also change the **Master Knowledge** attribute of the schema in the local DSA to that of the initial DSA (this is similar to giving the responsibility of mastering the schema to the initial DSA). Finally, the administrator would enter the initial DSA as a shadow in the local DSA so that the local DSA knows about it.

However, a copy of the initial DSA in the local DSA is not always required. In Figure 2-4, **dsa-osf** could be the initial DSA for the **OSF** administrative domain. The **dsa-munich** and **dsa-Boston** DSAs are normally connected to the initial DSA, **dsa-osf**. The DSA **dsa-doc** could be connected to the initial DSA **dsa-osf** or to **dsa-Boston**, which is not an initial DSA.

Figure 2–4. A Sample Tree with a First-Level DSA and an Initial DSA



To make a DSA part of an adminstrative domain where the default schema is not being used, the administrator carries out the steps mentioned in the previous example, as well as some additional steps. The administrator must copy the modified schema over to the local DSA, because the initial schema has changed. The administrator must also give the local DSA a shadow of the initial DSA and enter the local DSA and the initial DSA (including their PSAP addresses) in the DUA cache using the new schema structure.

2.4.2 First-Level DSA

A first-level object is an object under the root. If a DSA is master of a first-level object, it is called a *first-level DSA*. Typically, the first-level DSA is also the initial DSA for an administrative domain. However, as shown in the Figure 2-4, **dsa-usa** is the first-level DSA and **dsa-osf** is the initial DSA for the **OSF** administrative domain.

The first-level DSA is usually responsible for a country and for administering all the nodes in the DIT below a country. The first-level DSA is normally an administrative institution (such as the Deutsche Bundespost in Germany). The first-level DSA is responsible for the first node in the DIT (for example, **DE**, which is the designation for Germany). The first-level DSA in turn provides a node for one or more subordinate nodes. These subordinate nodes can be administrative domains responsible for managing the objects and holding the names unique in the directory system.

Normally, the first-level DSA knows all the other first-level DSAs in the directory system so that it can find all X.500 entries in a worldwide X.500 directory system.

Administrators must make sure that each DSA has at least a shadow entry of the DSA that masters the object in the next superior level in the DIT. If superior nodes are mastered by non-GDS DSAs, administrators also need to make sure that the shadows of these non-GDS DSAs are present as shadows in the local DSA.

2.4.3 Default DSA

The default DSA is the local or remote DSA contacted when the **Logon to the Default DSA** option is selected in the Logon Menu Mask (Mask 1) of the administration program **gdsditadm**. The administrator can specify one or more default DSAs by specifying a special attribute called a **DSA-Type** when entering a DSA object in the DUA cache. **DSA-Type** is described in Section 2.5.

2.5 DUA Cache

The administrator must add an object called **client** with the PSAP address of the client to the DUA cache to set up a connection to a remote DSA. The entries of all DSAs (including their PSAP addresses) that the DUA wants to establish direct connections to must also be added. Typically, the DUA cache must contain the name and the PSAP address of the local DSA.

However, the administrator can specify a special optional attribute called **DSA-Type. DSA-Type** enables the administrator to determine whether a particular DSA object entered in the cache is considered as the local DSA, the default DSA, or both the local and the default DSA.

''Local'' in this case means that the DSA of the directory system is in the same computer as the DUA. The local DUA does not need to establish a connection to the network in order to access it. The DUA sends its requests to the local DSA, which can access other remote DSAs and can be accessed by remote DUAs and DSAs.

''Default'' means that this DSA is the remote DSA that the administrator wants to contact when using the **Logon to the Default DSA** option from the User Identification Mask (Mask 1). To access this DSA, the DUA needs to establish a connection to the network.

''Default/local'' indicates that the local DSA is also the default DSA; normally, an administrator who has a client/server system needs to specify this DSA type.

Several default DSAs can be specified in the DUA cache. When the administrator logs into the default DSA, a connection is set up with the first available DSA in the list of DSAs entered in the cache. The administrator establishes priority according to the order in which the names of the default DSAs are entered in the cache.

The administrator can enter other DSA objects in the DUA without specifying the **DSA-Type** attribute. To set up a connection with one these remote DSAs, the administrator must select the **Logon to a Specific DSA** option in Mask 1. (See Chapter 7 for more information on Mask 1.)

Chapter 3

Developing a Configuration Plan

Before attempting to initialize and configure the directory service, the administrator must develop a configuration plan that is specific to the environment in which the directory service will run and that meets the requirements of the users who will access the directory service.

This chapter includes configuration worksheets that an administrator can use to develop a configuration plan. The list of worksheets includes the following:

- Cell worksheet
- ACL Schema worksheet
- ACL Object Entry worksheet
- Shadow Entries worksheet
- Client worksheet
- Client/Server worksheet
- Remote GDS and Non-GDS DSA worksheet

Appendix F contains a set of worksheets that administrators can copy for their own use. Administrators are not obliged to use every worksheet; they are provided to help to organize the information required to initialize and configure GDS and to make modifications at a later date.

After filling out the worksheets, the administrator is ready to perform a GDS configuration. The next chapter describes how to use this information to install and configure GDS.

In order to explain how a typical administrator works through the configuration process, this chapter refers to a hypothetical branch administrator, Branch Adminstrator 4, who is developing a plan to integrate a cell into GDS. Branch Administrator 4 is setting up the directory service for a branch office of a large corporation called the XYZ Corporation. The configuration consists of four machines: two clients and two client/servers. The two client/server machines have the DSAs **dsa-ops** and **dsa-employ**.

The initial DSA for the administrative domain containing **dsa-ops** is **dsa-HQ**. This DSA is administered by the corporate network administrator, Corporate Administrator 1, who is located at corporate headquarters in a different city than Branch Administrator 4.

The first-level DSA is administered by a global naming authority (described in the following section) and is called **dsa-us** in this example.

Developing a configuration plan involves the following steps:

- Specifying the namespace organizations
- Defining a cell in the directory
- Specifying ACLs
- Determing the number of machines
- Determining client/server addresses
- Defining non-GDS DSAs

3.1 Specifying the Namespace Organizations

This section covers the following topics:

- Registering with namespace organizations
- Determining the Distinguished Names of DSAs
- Determining the need to modify the standard schema
- Determining the need to modify directory IDs

3.1.1 Registering with Namespace Organizations

To enable a DCE cell to communicate with other cells in the global naming environment, an administrator needs to obtain a globally unique cell name. The name either exists already or is specially created for this purpose. To obtain a unique GDS cell name, an administrator contacts the other administrator in charge of the section of the Directory Information Tree (DIT) under which the cell is to be named. In the United States, the American National Standards Institute (ANSI) delegates X.500 names subordinate to the entry **/C=US**.

Administrators are encouraged to obtain a unique global name for a cell even if the cell does not initially use GDS to communicate with other cells. This enables the cell to participate subsequently in a global naming service.

In the sample configuration scenario, Corporate Administrator 1 in charge of network operations for XYZ Corporation applies to ANSI to reserve **XYZ** as a unique organization name. Later, as one of the first steps in establishing a cell, Branch Administrator 4 contacts Corporate Administrator 1 and asks to reserve the global name **Branch4**, which is used to create a cell entry **/C=US/O=XYZ/OU=Branch4** in GDS. Finally, Branch Administrator 4 again applies to Corporate Administrator 1 to reserve the names **dsa-employ** and **dsa-ops** to establish the entry names for the two DSAs:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**

If all directory requests that originate from **dsa-employ** and **dsa-ops** remain within XYZ Corporation, Branch Administrator 4 only needs to be registered with Corporate Administrator 1 (that is, within XYZ Corporation). In this case, Branch Administrator 4 makes sure that Corporate Administrator 1 at corporation headquarters creates a shadow entry of **dsa-employ** and **dsa-ops** on **dsa-HQ**. This way, Branch Administrator 4's DSAs can communicate with other DSAs through **dsa-HQ**.

Branch Administrator 4 also needs to make sure that a shadow of **dsa-HQ** is present on **dsa-ops** and **dsa-employ** so that these DSAs can access other DSAs in the XYZ corporation.

If Branch Administrator 4 expects the DSAs to participate in the directory service outside of XYZ Corporation, **dsa-employ** and **dsa-ops** must be known by DSAs outside of XYZ Corporation. If shadow entries of **dsa-employ** and **dsa-ops** exist on **dsa-HQ**, most DSAs can contact Branch

Administrator 4's DSAs through **dsa-HQ**. However, if Branch Administrator 4's users need to access these remote DSAs frequently, Branch Administrator 4 should consider contacting administrators of DSAs outside XYZ Corporation directly and arranging for mutual shadow entries to be present on their respective DSAs. This would reduce the overhead of referrals and chained requests over the network. In addition, **dsa-employ** and **dsa-ops** need to have a copy of **dsa-us** (the first-level DSA) to communicate worldwide.

3.1.2 Determining the Distinguished Names of DSAs

Each GDS entry is uniquely and unambiguously identified by a DN. The administrator must determine the DN of each DSA in the cell (or cells) that the administrator is responsible for in order to add these DNs to the directory.

For example, Branch Administrator 4 knows that the DN for XYZ Corporation is **/C=US/O=XYZ**, and that the DN of the cell is **/C=US/O=XYZ/OU=Branch4**. Branch Administrator 4 has applied for and received the names of the two DSAs: **dsa-employ** and **dsa-ops**. Therefore, the DN of the two DSAs in the cell are

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**

Branch Administrator 4 also needs to determine the DN of other DSAs (remote DSAs) in the network that users need to access directly, and the DN of **dsa-HQ** at XYZ corporate headquarters. Branch Administrator 4 can find out from Corporate Administrator 1 that the DN of **dsa-HQ** is **/C=US/O=XYZ/OU=mainbranch/CN=dsa/CN=dsa-HQ** and the DN of **dsa-us** is **/C=US/O=ANSI/OU=first-level-dsa/CN=dsa/CN=dsa-us**. Branch Administrator 4 sometimes has to contact the administrators of other remote DSAs directly to find out the DNs. However, if Branch Administrator 4's DSAs can connect to **dsa-HQ**, they can read the DNs of all DSAs in the network from the DIB.

3.1.3 Determining the Need to Modify the Standard Schema

GDS software is delivered with the GDS standard schema. It contains the directory classes and attributes specified in the X.500 standard, the GDS extensions and, optionally, the electronic mail package (X.400). An administrator sometimes needs to add object classes and attributes that are required by other applications to the schema.

For example, Branch Administrator 4 decides to install a new software package to monitor computer facilities. This package requires directory classes and attributes that are not part of the current schema. The new program is being implemented across XYZ Corporation. Corporate Administrator 1 has set up the directory so that responsibility for administering the schema resides at XYZ Corporation headquarters (**dsa-HQ** is the initial DSA and therefore masters the schema object). Branch Administrator 4 cannot make any changes to the schema without submitting a request to Corporate Administrator 1's office. Branch Administrator 4 contacts Corporate Administrator 1 and provides the values for the entries in the OCT, SRT, and AT for these new object classes and attributes. Corporate Administrator 1 adds the new object classes and attributes to the schema.

Corporate Administrator 1 also must use shadow administration functions to make sure that Branch Administrator 4 receives the schema changes in the shadow of the schema object.

3.1.4 Determining the Need to Modify Directory IDs

GDS allows an administrator to define multiple directory services by using directory IDs. This restricts access to an entire DIT. While this approach may be costly in terms of additional overhead, it is an effective means of insuring very tight security or segregation of information by function where required.

The XYZ Corporation can have more than one directory service system. For example, employee information can reside in one directory service system and customer data in another. The branch network administrator needs to find out from users if more than one directory service needs to be accessed.

3.2 Defining a Cell in the Directory

In order for a cell to be accessible in the global naming environment, it must have a global name that is defined in the DIT and is meaningful and usable from anywhere in the DCE naming environment.

The global name for the branch network administrator's cell, which is obtained from the corporate network administrator at corporate headquarters, is **/.../C=US/O=XYZ/OU=Branch4**.

Branch Administrator 4 creates an entry for the cell in the directory by using object administration. In GDS, additional cell information is contained in two attributes: **CDS-Cell** and **CDS-Replica**. The GDS object administration program provides Mask 21 and Mask 22 for administrators to enter values for these attributes. (See Chapter 8 for more information about object administration masks.)

CDS-Cell and **CDS-Replica** attributes have the following fields associated with them:

Namespace UUID

The Universal Unique Identifier (UUID) of the CDS namespace. This field is required to resolve ambiguity between CDS namespaces when a server manages more than one clearinghouse, and the clearinghouses are in different namespaces.

A CDS UUID consists of 16 hex digit pairs represented as 8 hexadecimal digits followed by a hyphen, 3 groups of 4 hexadecimal digits separated by hyphens, a hyphen and 12 hexadecimal digits (for example, 01234567-89ab-cdef-0123-456789abcdef).

Root Dir UUID

Used to form the resolved and unresolved names in a CDS progress record along with **Root Dir Name**.

Root Dir Name

Used to form the unresolved and resolved names in a CDS progress record along with **Root Dir UUID**. These parameters are only required when multiple cells are contained in the CDS namespace.

Replica Type

Specifies whether a replica is a master (modifiable) replica or a read-only replica. (A master replica can be changed to read-only).

Clearinghouse UUID

The clearinghouse UUID which never changes. The clearinghouse name can change; the clearinghouse UUID is used to check the validity of the clearinghouse.

Clearinghouse Name

Contains the full name of the clearinghouse in which a replica is stored. This name is the name of a naming attribute that contains information on the last known address of the clearinghouse. This information enables the creation of an RPC binding to the server that maintains the clearinghouse.

Tower

The tower set of the server that maintains the clearinghouse. A CDS tower contains addressing information and information on protocols supported by the clearinghouse server. The format of the tower value, which is the same format as a substring of the RPC string binding, is as follows:

protseq:netaddr

An administrator uses the **cdscp show cell** command to generate this information. This means that an administrator does not require detailed knowledge of data formats or CDS concepts.

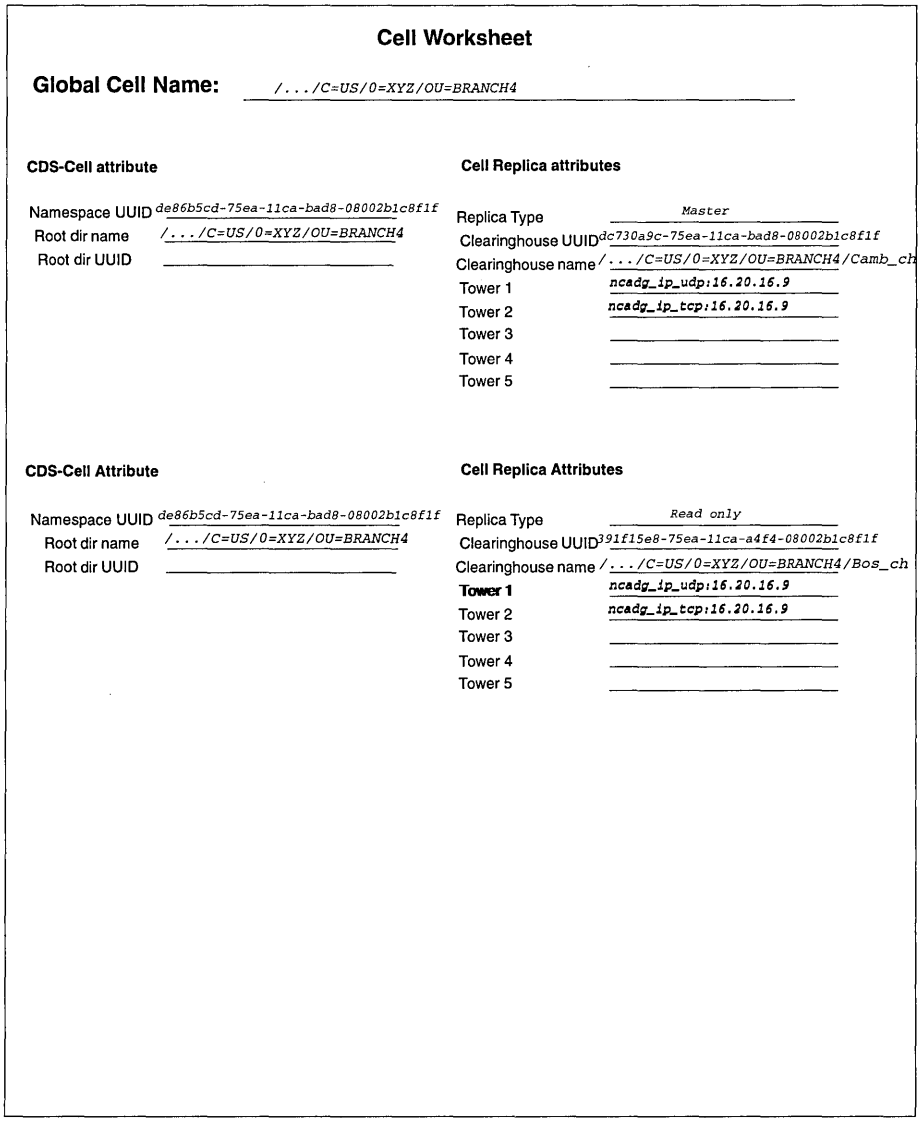
The following is a sample **show cell** command and the resulting GDS-formatted output for a cell named **/.../C=US/O=XYZ/OU=Branch4**:

```
cdscp> show cell /.../C=US/O=XYZ/OU=Branch4 as gds

            SHOW
            CELL    /.../C=US/O=XYZ/OU=Branch4
              AT    1991-09-18-17:17:23
Namespace UUID = de86b5cd-75ea-11ca-bad8-08002b1c8f1f
   Chouse UUID = dc730a9c-75ea-11ca-bad8-08002b1c8f1f
   Chouse Name = /.../C=US/O=XYZ/OU=Branch4/Camb_ch
 Replica Type = Master
     Tower 1  = ncadg_ip_udp:16.20.16.9
     Tower 2  = ncacn_ip_tcp:16.20.16.9
```

```
Namespace UUID = de86b5cd-75ea-11ca-bad8-08002b1c8f1f
   Chouse UUID = 391f15e8-75ef-11ca-a4f4-08002b1c8f1f
   Chouse Name = /.../C=US/O=XYZ/OU=Branch4/Bos_ch
  Replica Type = Readonly
      Tower 1  = ncadg_ip_udp:16.20.16.9
      Tower 2  = ncacn_ip_tcp:16.20.16.9
```

Branch Administrator 4 uses the information in the output from the **show cell** command to fill in the Cell worksheet with appropriate information for the cell as shown in Figure 3-1.

Figure 3–1. The Branch Network Administrator's Cell Worksheet



Cell Worksheet

Global Cell Name: /.../C=US/O=XYZ/OU=BRANCH4

CDS-Cell attribute

Namespace UUID de86b5cd-75ea-11ca-bad8-08002b1c8f1f
Root dir name /.../C=US/O=XYZ/OU=BRANCH4
Root dir UUID

Cell Replica attributes

Replica Type Master
Clearinghouse UUID dc730a9c-75ea-11ca-bad8-08002b1c8f1f
Clearinghouse name /.../C=US/O=XYZ/OU=BRANCH4/Camb_ch
Tower 1 ncadg_ip_udp:16.20.16.9
Tower 2 ncadg_ip_tcp:16.20.16.9
Tower 3
Tower 4
Tower 5

CDS-Cell Attribute

Namespace UUID de86b5cd-75ea-11ca-bad8-08002b1c8f1f
Root dir name /.../C=US/O=XYZ/OU=BRANCH4
Root dir UUID

Cell Replica Attributes

Replica Type Read only
Clearinghouse UUID 391f15e8-75ea-11ca-a4f4-08002b1c8f1f
Clearinghouse name /.../C=US/O=XYZ/OU=BRANCH4/Bos_ch
Tower 1 ncadg_ip_udp:16.20.16.9
Tower 2 ncadg_ip_tcp:16.20.16.9
Tower 3
Tower 4
Tower 5

3.3 Specifying ACLs

Branch Administrator 4 needs to determine the access protection required for each object in the directory at the attribute level to ensure that unauthorized users cannot read or modify attributes requiring some form of restricted access.

Each attribute of an object is defined in the schema as having one of the following access classes:

- Public
- Standard
- Sensitive

The administrator can change the access class of specific attributes by using schema administration functions (described in Chapter 9).

Each object in the directory has an ACL attribute that contains values for the five access classes:

- Modify Public
- Read Standard
- Modify Standard
- Read Sensitive
- Modify Sensitive

Modify Sensitive is the most restrictive category. Modify Public is the least restrictive of the five categories. Read Public is, by definition, available to every user of the directory service.

An administrator must also determine the access that users require for object entries. The administrator assigns directory users to the specific access classes by entering their DNs in the mask provided in the object administration program (described in Chapter 8).

For example, the Personnel Department is responsible for the public and private data associated with each employee of the branch office. The personnel manager has given Branch Administrator 4 a list of information that should be made available to all employees at the branch office. This list includes telephone numbers, fax numbers, and so on. The personnel manager requests that only certain people have the ability to modify telephone and

fax numbers in the directory. Another list contains information such as salaries, home addresses, and other personal data that the personnel manager wants restricted so that the data can be read but not modified.

Figure 3-2 shows the relationship between access classes as defined in the schema and how user access class assignments affect access to a specific object entry. **Fax-Telephone-Number**, **Telephone-Number**, **Street-Address**, and **Access-Control-List** are attributes of the object entry **/C=US/O=US/OU=Branch4/CN=Jack Jones**. The access classes for the **Fax-Telephone-Number**, **Telephone-Number**, **Street-Address**, and **Access-Control-List** attributes have been defined in the schema as **PUBLIC**, **PUBLIC**, **SENSITIVE**, and **SENSITIVE** respectively. The object entry has the values of its **ACL** attribute defined for Mary Smith, John Dulles, and Jack Jones.

As shown in Figure 3-2, Mary Smith has **MODIFY PUBLIC** and **READ SENSITIVE** access. This means that

- She can read and modify the **Fax-Telephone-Number** and **Telephone-Number** attributes.
- She can read, but not modify, the **Street-Address** and **Access-Control-List** attributes.

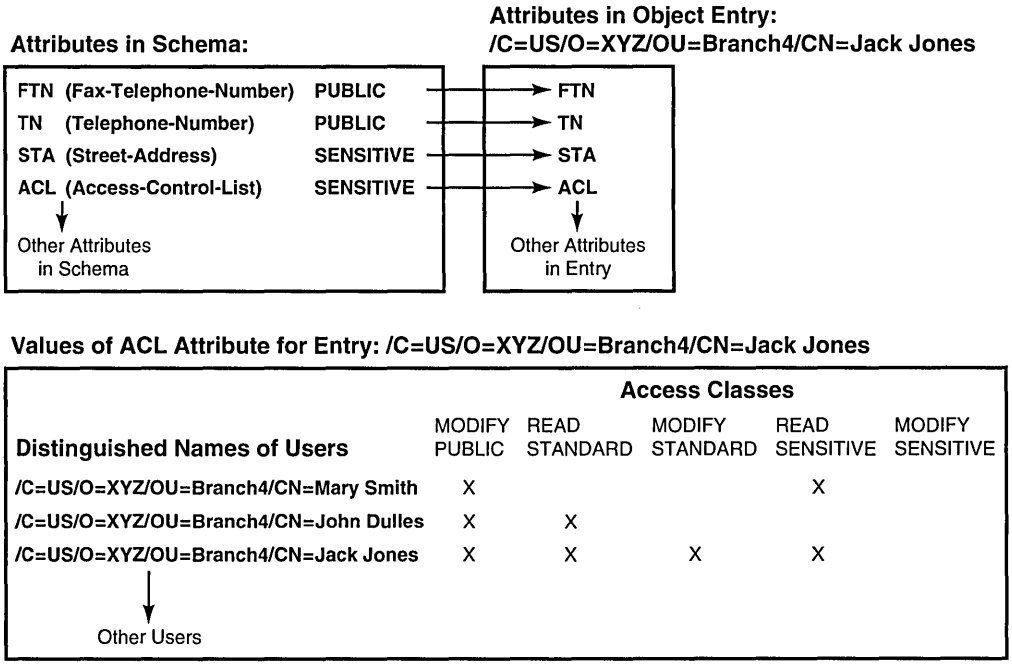
John Dulles has **MODIFY PUBLIC** and **READ STANDARD** access. This means that

- He can read and modify **Fax-Telephone-Number** and **Telephone-Number** attributes.
- He cannot read or modify the **Street-Address** and **Access-Control-List** attributes.

Jack Jones has **MODIFY PUBLIC**, **READ STANDARD**, **MODIFY STANDARD**, and **READ SENSITIVE** access. This means that

- He can read and modify **Fax-Telephone-Number** and **Telephone-Number** attributes.
- He can read, but not modify, the **Street-Address** and **Access-Control-List** attributes.

Figure 3–2. Sample ACLs for Four Attributes



For this entry Mary Smith:

- Can read and modify the **FTN** and **TN** attributes.
- Can read but not modify the **STA** and **ACL** attributes.

For this entry John Dulles:

- Can read and modify the **FTN** and **TN** attributes.
- Cannot read or modify the **STA** and **ACL** attributes.

For this entry Jack Jones:

- Can read and modify the **FTN** and **TN** attributes.
- Can read but not modify the **STA** and **ACL** attributes.

When setting up the access type for each user, Branch Administrator 4 considers the requirements of those people responsible for managing specific information in the directory (such as the Personnel Manager).

Branch Administrator 4 defines the access rights for specific users, using Mask 6b of the Object Administration operations to add them to the directory. There are two types of ACL worksheets:

- ACL Schema Worksheet
- ACL Object Entry Worksheet

ACL Schema Worksheet

The ACL Schema worksheet contains a list of the access classes that the administrator plans to assign for each attribute requiring a change from the original value in the GDS standard schema.

Figure 3-3 shows a sample ACL Schema worksheet, which contains the attributes and access class values that the branch network administrator wants to assign. **Surname** and **Telephone-Number** are **PUBLIC** so that all users can read them; however, Branch Administrator 4 also wants the ability to restrict users from modifying them for particular entries. Attributes (such as **User-Password**) that Branch Administrator 4 does not want users to be able to read or modify are assigned to the **SENSITIVE** access category.

Branch Administrator 4 uses schema administration operations to make all the changes to the schema.

Figure 3–3. Sample ACL Schema Worksheet

ACL Schema Worksheet

Attibute Type	Access Class		
	PUBLIC	STANDARD	SENSITIVE
Common-Name	X		
Surname	X		
Serial-Number	X		
Telephone-Number	X		
Street-Address		X	
Presentation-Address			X
Structure-Rule-Table			X
Time-Stamp			X
CDS-Cell			X
CDS-Replica			X
User-Password			X

ACL Object Entry Worksheet

The ACL Object Entry worksheet contains a list of the DNs and access classes assigned to each user for each object entry that requires specific access to its attributes. The information in the worksheet specifies the values of the ACL attribute for a specific object. The **Interpretation** heading indicates whether the single object or all objects in the subtree below it are included in the access assignment.

Figure 3-4 shows how the branch network administrator has filled in a sample ACL Object Entry worksheet. For the entry **/C=US/O=XYZ/OU=Branch4/CN=Jack Jones**:

- Mary Smith has **MODIFY PUBLIC** and **READ SENSITIVE** access.
- John Dulles has **MODIFY PUBLIC** and **READ STANDARD** access.
- Jack Jones has access in all five categories.

For the second object entry **/C=US/L=cambridge/CN=Paul Lewis**, all objects below **/C=US/O=XYZ/OU=BRANCH4** have **MODIFY PUBLIC** to **READ SENSITIVE** access. Only Branch Administrator 4 has **MODIFY SENSITIVE** access.

Figure 3–4. Sample ACL Object Entry Worksheet

ACL Object Entry Worksheet

Directory entry: */C=US/O=XYZ/OU=Branch4/CN=James Jones*

Access Class	Distinguished Name of User	Interpretation (single object or subtree)
modify public	*/C=US/O=XYZ/OU=Branch4/CN=Mary Smith*	*single object*
	/C=US/O=XYZ/OU=Branch4/CN=John Dulles	*single object*
	/C=US/O=XYZ/OU=Branch4/CN=James Jones	*single object*
read standard	*/C=US/O=XYZ/OU=Branch4/CN=John Dulles*	*single object*
	/C=US/O=XYZ/OU=Branch4/CN=James Jones	*single object*
modify standard	*/C=US/O=XYZ/OU=Branch4/CN=James Jones*	*single object*
read sensitive	*/C=US/O=XYZ/OU=Branch4/CN=Mary Smith*	*single object*
	/C=US/O=XYZ/OU=Branch4/CN=James Jones	*single object*
modify sensitive	*/C=US/O=XYZ/OU=Branch4/CN=James Jones*	*single object*

Directory entry: */C=US/L=Cambridge/CN=Paul Lewis*

Access Class	Distinguished Name of User	Interpretation (single object or subtree)
modify public	*/C=US/O=XYZ/OU=Branch4*	*subtree*
read standard	*/C=US/O=XYZ/OU=Branch4*	*subtree*
modify standard	*/C=US/O=XYZ/OU=Branch4*	*subtree*
read sensitive	*/C=US/O=XYZ/OU=Branch4*	*subtree*
modify sensitive	*/C=US/O=XYZ/OU=Branch4/CN=Joe*	*single object*

3.3.1 Directory IDs

Directory IDs are another means of segregating information by restricting access to a specific group of users. They also can be used if two applications need a completely different tree structure and completely different attributes in the DIT.

3.3.2 Setting ACLs for the Schema

After GDS is configured, the ACL of the default schema has no access rights. This means that every user, including the anonymous user, has read and write access to all attributes in the schema. The branch network administrator needs to ask the corporate network administrator to change the ACL of the default schema (the corporate network administrator is responsible for the initial DSA **dsa-HQ**) to only permit read and write access to appropriate users. Applications often require read access to the schema. The corporate network administrator needs to be the only one with write access to the schema.

3.4 Determining the Number of Machines

This section provides some guidelines to help an administrator gather information and make decisions on various aspects of client and client/server machines, DUA caches on each machine, and different types of DSAs.

3.4.1 Clients and Servers

An administrator must determine which of the machines in a cell are to be clients and which are to be client/servers. The number of client machines in a cell depends on the number of users who need to access a local DSA at the same time. If this number is high, it is worthwhile to include more than one client in the configuration plan. The number of client/server machines depends on the number of DSAs that are started when the directory service

is activated. Normally, the optimal number of DSAs is two. However, more DSAs are required if one machine has to reply to a number of requests at the same time.

For example, Branch Administrator 4 determines that the configuration for the branch office will consist of two client/servers and two clients. Branch Administrator 4 also wants to have one DSA running on each machine so that, if one goes down, users will have uninterrupted access to other DSAs in the network. The two DSAs are called **dsa-employ** and **dsa-ops**. The **dsa-employ** DSA masters all employee records for the branch; **dsa-ops** masters other data related to the daily operations of the branch.

3.4.2 Initial DSA

An initial DSA masters the schema object. An administrator must make sure that there is a shadow of the schema object on each DSA that is a member of an initial DSA's administrative domain.

The initial DSA in Branch Administrator 4's network is **dsa-HQ**, which is located at XYZ corporate headquarters. This DSA is administered by Corporate Administrator 1, who is responsible for maintaining the schema. Branch Administrator 4 must make sure that **dsa-employ** and **dsa-ops** contain shadows of the schema object so that they can become members of the administrative domain of **dsa-HQ**.

3.4.3 First-Level DSA

First-level DSAs are usually maintained by naming authorities at the national level (such as the Deutsche Bundespost in Germany and ANSI in the US). Typically, administrators only need to ensure that the first-level DSAs in the network are either known to at least one of the DSAs under their care or can be accessed by referral through other DSAs in the network. In the branch network administrator's case, **dsa-employ** or **dsa-ops** needs to have a shadow entry of the first-level DSA because, in the case of global requests, **dsa-employ** or **dsa-ops** can try to chain an operation to the first-level DSA.

3.4.4 Master and Shadow Entries

An administrator must decide which DSAs will contain master entries and which will contain shadow entries. If the most up-to-date information is required, the user needs to obtain it directly from the master. In this case, the administrator must keep the master entry on a machine close to users (avoiding network access where possible). If users at other sites require frequent access, but the accuracy of the retrieved data is not time-critical, shadows can be created at these other sites.

An administrator also needs to determine how often shadows are updated. The frequency depends on user requirements and the types of applications that need access to the directory.

For example, employee phone numbers do not change frequently. If employees within a small company do not change addresses that often, the shadow update frequency could be low; for example, daily or even weekly. However, if XYZ Corporation is experiencing explosive growth, and there are hundreds of thousands of employees worldwide, shadowing would be required on a more frequent basis.

As a general rule, most entries in GDS do not require a high update frequency. Normally, shadow entries should be updated once daily. However, if the directory service is used as a database, data tends to change more frequently.

Figure 3-5 shows a sample Shadow Update worksheet partially filled out by Branch Administrator 4. The worksheet includes columns for the DN of the object entry, the update frequency, and the update times. The possible values for the update frequency are **HIGH**, **MEDIUM**, and **LOW**. The values of the update times depend on the update frequency (update times are in minutes for **HIGH**, hours for **MEDIUM**, and a specific hour in the range of days for **LOW**).

Figure 3–5. The Branch Network Administrator's Partial Shadow Worksheet

Shadow Update Worksheet

Initiator: /C=US/O=XYZ/OU=BRANCH4/CN=dsa/CN=dsa-ops

Target (shadow DSA): /C=US/O=XYZ/OU=BRANCH4/CN=dsa/CN=dsa-employ

Distinguished Names of Entries	*Interpretation (object/subtree)*	*Update Frequency (HIGH, LOW, or MEDIUM)*	*Update Time (minutes, hours, or days)*
/C=US/O=XYZ/OU=BRANCH4/CN=James Jones	object	high	every 15 minutes
/C=US/O=XYZ/OU=BRANCH4/CN=Al Smith	object	high	every 15 minutes

3.4.5 Default DSAs

When entering a DSA object in the DUA cache, an administrator can specify a value for the **DSA-Type** attribute. The value of this attribute can be the value of **local**, **default**, or **default/local**.

For each of the clients, an administrator must determine which DSA is the local or default DSA or both. The default/local DSA (DSA-Type **default/local**) is typically the one located on the nearest machine in the network. More than one default DSA can be entered in the DUA cache (DSA-Type **default**). This provides an alternative path to the network if one or more of the other default DSAs cannot be reached.

In addition, the administrator needs to determine which other DSAs (remote DSAS) the DUA needs to contact directly, and enter their names and PSAP addresses in the cache (the value for DSA-Type is left blank). DSAs that are neither default nor local do not have a value assigned for DSA-Type in the DUA cache.

On the Client worksheet in Figure 3-6, Branch Administrator 4 enters the DNs of **dsa-employ** and **dsa-ops** as default DSAs on **client1** and **client2**.

On the Client/Server worksheet in Figure 3-7, Branch Administrator 4 enters **dsa-ops** as the default/local DSA and **dsa-employ** as a default DSA on **clientserver1** (the machine on which **dsa-ops** resides). Branch Administrator 4 also enters **dsa-employ** as the default/local DSA and **dsa-ops** as the default DSA on **clientserver2** (the machine on which **dsa-employ** resides).

The branch network administrator also defines **dsa-HQ** as another default DSA for **client1** so that, if **dsa-employ** or **dsa-ops** is not accessible, other DSAs can be contacted through **dsa-HQ**. The branch network administrator also defines **dsa-us** as another default DSA in the DUA cache of **client2**, because it is important that **client2** be able to access DSAs outside of the XYZ Corporation's namespace.

The branch network administrator enters the DN of **dsa-HQ** as a remote GDS DSA on the worksheet for **client1**.

3.5 Determining Client/Server Addresses

An administrator needs to know the PSAP addresses of server machines and client stubs in order to integrate them into the directory system. As discussed in Chapter 2, PSAP addresses are composed of one or more NSAP addresses and associated Transport selectors. (Presentation and Session selectors are not used by DCE.)

Typically, the server and client stubs are already part of a network and have been assigned a unique NSAP address by the appropriate naming authority. If this is not the case, the administrator must apply for a unique NSAP address. (See Appendix D for more information on how to apply for an NSAP address.)

As shown in Figures 3-6 and 3-7 Branch Administrator 4 has obtained the network addressing information for the two clients and two client/servers and entered it on the Client and Client/Server worksheets.

The last column on the Client worksheet lists the general type of DSA. This is for information purposes only. The values entered in this column do not correspond to any input data required to configure GDS.

Figure 3–6. Sample Client Worksheet

Client Worksheet

Global Cell Name: /.../C=US/O=XYZ/Branch4

Name of client machine: client1

PSAP address (client stub):

NSAP address TCP/IP!internet=192.35.18.1+port=21010

Transport procotol TCP/IP

T-Selector client P-Selector _____ S-Selector _____

DUA Cache Information

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ	default/local	remote GDS
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops	default	remote GDS
/C=US/O=XYZ/OU=mainbranch/CN=dsa/CN=dsa-HQ	default	remote GDS

Global Cell Name: /.../C=US/O=XYZ/Branch4

Name of client machine: client2

PSAP address (client stub):

NSAP address TCP/IP!internet=192.35.18.2+port=21014

Transport procotol TCP/IP

T-Selector client P-Selector _____ S-Selector _____

DUA Cache Information

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops	default/local	remote GDS
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ	default	remote GDS
/C=US/O=ANSI/OU=first-level dsa/CN=dsa/CN=dsa-US	default	first-level

Figure 3–7. Sample Client/Server Worksheet

Client/Server Worksheet

Global Cell Name: /.../C=US/O=XYZ/Branch4

Name of client/server machine: client/server1

Distinguished name of DSA: /C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops

PSAP address:

NSAP address: TCP/IP!internet=192.35.18.4+port=21018

Transport procotol: TCP/IP

T-Selector: server P-Selector: S-Selector:

DUA Cache Information

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops	default/local	GDS
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ	default	remote GDS
/C=US/O=XYZ/OU=mainbranch/CN=dsa/CN=dsa-HQ	default	remote GDS

Global Cell Name: /.../C=US/O=XYZ/Branch4

Name of client/serve machine: client/server2

Distinguished name of DSA: /C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ

PSAP address:

NSAP address: TCP/IP!internet=192.35.18.6+port=21020

Transport procotol: TCP/IP

T-Selector: server P-Selector: S-Selector:

DUA Cache Information

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ	default/local	GDS
/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops	default	remote GDS
/C=US/O=ANSI/OU=first-level dsa/CN=dsa/CN=dsa-US	default	first-level

3.6 Defining Remote GDS and Non-GDS DSAs

An administrator needs to determine the DN, DSA type, PSAP address, and transport protocol information for remote DSAs that users require access to.

Figure 3-8 shows how Branch Administrator 4 enters this information for each of the remote DSAs **dsa-remote1** and **dsa-remote2** on the GDS Remote and Non-GDS worksheet. This information is used to create the required entry in the DUA caches of the relevant client and client/server machines.

Figure 3-8 also shows how Branch Administrator 4 enters this information for the non-GDS DSA **dsa-remote-non-GDS**. This information is used to create the required entry in the DUA caches of the relevant client and client/server machines.

Figure 3–8. Sample GDS Remote and Non-GDS DSA Worksheet

GDS Remote and Non-GDS DSA Worksheet

Distinguished name of DSA: */C=US/O=XYZ/OU=Branch2/CN=dsa/CN=dsa-remote1* DSA type *remote GDS*

PSAP address (DSA)

NSAP address *IBMLAN!ethernet=800148101D3*

Transport protocol *OSI-LAN*

T-Selector *server* P-Selector _____ S-Selector _____

Distinguished name of DSA: */C=US/O=XYZ/OU=Branch3/CN=dsa/CN=dsa-remote2* DSA type *remote GDS*

PSAP address (DSA)

NSAP address *IBMLAN!ethernet=800148101F4*

Transport protocol *OSI-LAN*

T-Selector *server* P-Selector _____ S-Selector _____

Distinguished name of DSA: */C=US/O=ANSI/OU=first-level dsa/CN=dsa/CN=dsa-US* DSA type *first-level, remote non-GDS*

PSAP address (DSA)

NSAP address *TCP/IP!internet=192.40.20.4+port=22012*

Transport protocol *TCP/IP*

T-Selector *server* P-Selector _____ S-Selector _____

Distinguished name of DSA: _____ DSA type _____

PSAP address (DSA)

NSAP address _____

Transport protocol _____

T-Selector _____ P-Selector _____ S-Selector _____

Chapter 4

Overview of the GDS Administration Tools

This chapter is an overview of the GDS administration tools. It provides a brief description of the commands used to invoke GDS and a brief description of administration functions. Several administration functions related to maintaining the system are described in greater detail. This chapter also describes the structure and purpose of masks and provides an overview of function key usage.

GDS can be started by using the following commands:

gdssysadm Supports administration of the local GDS installation, such as configuration, server, activation, and backup

gdsditadm Supports administration of the contents of a GDS database and the local cache

gdscacheadm

Supports administration of the local DUA cache (only necessary on client systems without **gdsditadm**)

The GDS commands can be used either in dialog or batch mode. Dialog mode is used by an administrator to enter data interactively from a terminal using numbered screens called *masks*, which are provided as part of the GDS software. The administrator uses batch mode to run shell scripts that have been created to automate administrative procedures. (See the *OSF*

DCE Administration Reference for more information on how to use GDS commands in shell scripts.)

The GDS command-line options are described in the *OSF DCE Administration Reference*.

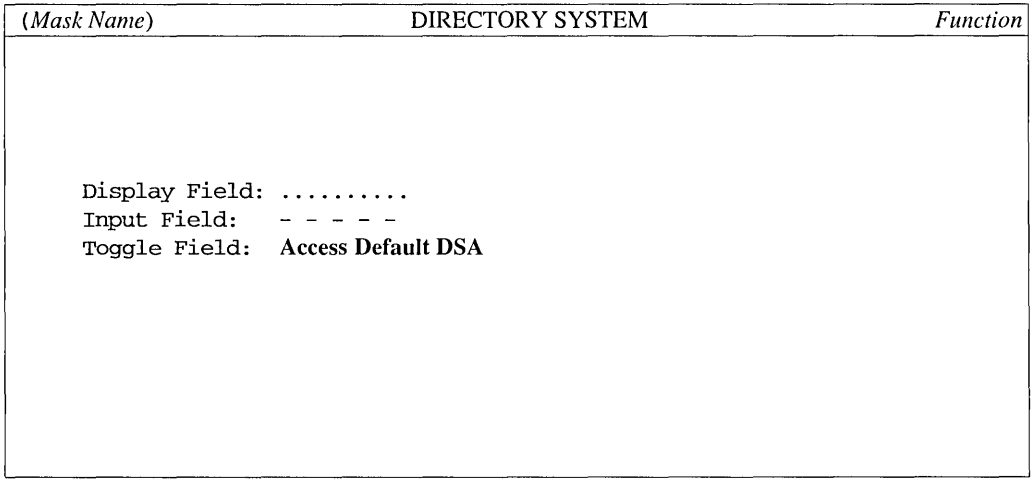
4.1 Mask Structure

As shown in Figure 4-1, the first line in every mask contains:

- The current mask name
- The current function

The other mask lines contain mask-dependent protected and unprotected mask fields. Unprotected mask fields are modifiable; protected fields are not.

Figure 4–1. General Mask Structure



This manual represents mask fields as follows:

Text — Protected display field for displaying function-dependent program variables such as in *Mask Name* or *Function* in Figure 4-1.)

.....	Protected display field for outputting directory data.
- - - -	Unprotected field for entering input (may be preset).
text	Selection field (toggle field) where specified options can be selected by pressing the space bar. (Preset options are displayed in the mask.)

Note: In GDS, selection fields are displayed in bold for highlighting purposes only. They are displayed normally on the screen.

4.2 The gdssysadm Command

To administer the directory system, the administrator calls the **gdssysadm** process to invoke GDS system administration. When **gdssysadm** is invoked, GDS presents the GDS Main Menu (Menu Mask, Part 1) as shown in Figure 4-2.

Figure 4–2. Menu Mask (Part 1)

```
(diradm)                          DIRECTORY SYSTEM

  a - Administration of the directory information tree

  c - Configuration of a directory system

  b - Activation of a directory system installation

  d - Deactivation of a directory system installation

  s - Saving of local data to diskette/tape/file

  r - Restoring of saved data from diskette/tape/file

  f - Further functions

        Your selection ! >
```

The user can administer the GDS system with the following functions:

- **a - Administration of the directory information tree/cache**

  Provides access to the DSA (see Chapter 7) and then provides the Object Administration, Schema Administration, Shadow Administration, and Subtree Administration functions. Alternatively, it provides access to the DUA cache and then provides the Object Administration and Cache Update functions.

- **c - Configuration of a directory system**

  Determines the type of configuration of the directory service and the number of server and client processes to be activated, and allows an administrator to create, delete, modify, and display configuration data.

- **b - Activation of a directory system installation**

  Activates the directory service by starting its background processes.

- **d - Deactivation of a directory system installation**

  Deactivates the directory service by terminating its background processes.

- **s - Saving of local data to diskette/tape/file**

  Saves the local data files of a directory system to diskette, tape, or file.

- **r - Restoring of saved data from diskette/tape/file**

  Restores the saved data of a directory system from a diskette, tape, or file.

- **f - Further functions**

  Displays the second part of the menu mask

If **f - Further functions** is selected in part 1 of the menu mask, GDS displays part 2 of the menu mask, as shown in the Figure 4-3.

Figure 4–3. Menu Mask (Part 2)

```
(diradm)                          DIRECTORY SYSTEM


  i - Display of directory system status information

  l - Activation of the 'trace' system

  t - Deactivation of the 'trace' system




Your selection ! >
```

The following options are available from the Menu Mask (Part 2):

- **i - Display of directory system status information**

  Displays whether the directory system is active or inactive, which processes are available and how many, and whether the trace system is active or inactive.

- **l - Activation of the 'trace' system**

  Starts the trace system for logging the directory processes.

- **t - Deactivation of the 'trace' system**

  Ends the trace system.

4.2.1 Saving Local Data to Diskette/Tape/File

If **s** is selected in the Menu Mask Part 1 (see Figure 4-2), the mask shown in Figure 4-4 is displayed. Use this mask to save the data of a local directory system to diskette, tape, or file.

Figure 4–4. Mask for Saving the Database of a Local Directory System

```
(savepar)                    DIRECTORY SYSTEM                    Save Data


For which directory ID do you wish to
save the local data ? [1-20]: 1

On which medium do you wish to save the local data ? Diskette

Name of the file: - - - - -

Security password, if required: - - - - -

Do you wish to format the media ?: NO
```

The mask displays the following fields:

For which directory ID do you wish to save the local data ? [1-20]:

Enter the directory ID, between **1** and **20**, of the directory system whose database you wish to save.

On which medium do you wish to save the local data ?

Select one of the following values. Toggle by pressing the space bar.

- **Diskette**
- **Tape**
- **File**

Name of the file:

Enter the name of the file to which the directory data is to be saved. (This field is only displayed if **File** is selected in the previous field.) The filename can be either an absolute or a relative filename. If it is a relative filename, the file is created in the subdirectory as specified in the **TARPATH** variable of the **dirparam** file (in **/opt/dcelocal/var/adm/directory/gds/conf**). Note that, if GDS is deinstalled, all the saved files will be lost if they are stored in subdirectories of **/opt/dcelocal**. (The default value of **TARPATH** is **/opt/dcelocal/var/adm/directory/gds/adm.**)

Security password, if required:

If you want to protect the data files in your directory system with a password, enter a password up to 10 characters long.

Do you wish to format the media ?

Select **YES** if you wish to format the data media before saving or select **NO** if you do not wish to format the data media. Toggle by pressing the space bar.

The saving process includes all the local data files (local DSA data, DUA cache data) belonging to the directory system.

If further diskettes are required, they are requested by the save procedure. If one of the data files exceeds the physical disk size, a tape must be used.

4.2.2 Restoring Saved Data from Diskette/Tape/File

If the **r** function is selected in the Menu Mask Part 1 (see Figure 4-2), the mask shown in Figure 4-5 is displayed. Use this mask to restore saved data from diskette or tape.

Figure 4–5. Mask for Restoring Data Saved from Diskette/Tape

```
(respar)                        DIRECTORY SYSTEM                        Restore Data


  For which directory ID do you wish to restore the
  local data ? [1-20]: 1

  From which medium should the data be read ?: Diskette

  Name of the file: - - - - -

  Security password: - - - - -

  Attention: Your existing data will be overwritten!!!
```

The mask displays the following fields:

For which directory ID do you wish to restore the local data ? [1-20]

Enter the directory ID, between **1** and **20**, of the local data you want to restore.

From which medium should the data be read ?

Select one the following values. Toggle by pressing the space bar.

- **Diskette**
- **Tape**
- **File**

Name of the file:

Enter the name of the file from which the directory data is to be restored. (This field is only displayed if **File** is selected as the medium.) The filename can be either an absolute or a relative filename. If it is a relative filename, the file is read from the subdirectory as specified in the **TARPATH** variable

of the **dirparam** file (in **/opt/dcelocal/var/adm/directory/gds/conf**). The default value of **TARPATH** is **opt/dcelocal/var/adm/directory/gds/adm**.

Security password:

Enter the password used when the local data was saved.

The system then asks you to insert the diskette or tape from which the data is to be read. The restoring process restores all the local data files belonging to the directory system. During the restoring process, all data in the DSA and in the DUA cache is overwritten. If additional diskettes were used when the data was saved to disk, these are requested by the restore procedure.

4.2.3 Displaying of Directory System Status Information

If **i** is selected in the Menu Mask Part 2 (see Figure 4-3), the mask shown in Figure 4-6 is displayed.

Figure 4–6. Mask for Displaying the Directory System Status Information

```
(Info)                          DIRECTORY SYSTEM                Status Information


 The directory system is active (existing processes ->):

              1  DUA cache process
              1  C-stub process
              1  S-stub process(es)
              5  DSA process(es)
              1  IPC monitoring process



 Status of the 'trace' system: active
```

To continue please press **<Return>**.

This mask displays the following information:

- Whether the directory system is active or inactive
- Which and how many processes are available
- Whether the trace system is active or inactive

Note: If a process that is to run is not displayed, the reason for the abort is documented in the corresponding log file.

4.2.4 Activating the trace System

If **1** is selected in the Menu Mask Part 2 (see Figure 4-3), the trace system for logging the directory processes is started.

4.2.5 Deactivating the trace System

If the **t** function is selected in the Menu Mask Part 2 (see Figure 4-3), the trace system for logging the directory processes is ended.

4.3 The gdsditadm Command

The **gdsditadm** command invokes directory database administration. If the administrator is connecting to a DSA, the command can be used to invoke Object, Schema, Shadow, and Subtree Administration directly. If the administrator is connecting to the DUA cache, the command can only be used to invoke Object Administration or Cache Update Administration directly.

4.3.1 Object Administration

Administrators manage objects and their attributes. Although both master and shadow entries can be accessed through the GDS adminstration tools, it is recommended that shadows be managed exclusively with the Shadow Administration functions. The Shadow Administration functions guarantee consistency in the DIT by updating shadows periodically using the master information. Because the Shadow Administration functions are not standardized by the X.500 standards, it is sometimes necessary to administer shadows with the Object Administration functions (for example, non-GDS DSAs).

Table 4-1 shows functions that are supported for logging into DSAs.

Table 4–1. Object Administration Functions

Function	Meaning
Add Object	Adds a new object with attributes and access rights.
Remove Object	Deletes an object.
Display Objects (Global Master Info)	Displays all master information on the selected objects. This operation contacts all DSAs that are involved by the query. The result is only complete if every DSA involved is available.
Display Objects (Entries in Current DSA)	Displays only the entries in the current DSA for selected objects; that is, master and shadow entries stored in this DSA.
Add Attributes	Adds one or more attributes for an object with a name and value. The attributes to be added must be defined in the Attribute Table (AT).
Delete Attributes	Deletes one or more attributes of an object.
Modify Attribute	Changes the attribute value of an object. Single value attributes can be modified and recurring attribute values can be added, modified, and deleted.
Add Alias	Adds an alias for an object.
Modify RDN	Changes the RDN of an object.

4.3.2 Schema Administration

The functions described in Table 4-2, which are supported by **gdsditadm**, allow the user to manage the object classes entered in the Object Class Table (OCT), the attribute types entered in the Attribute Table (AT), and the Structure Rules entered in the Structure Rule Table (SRT).

This is similar to the administration of a database schema.

Table 4–2. Schema Administration Functions

Function	Meaning
Display OCT	Displays the elements of the OCT
Add OCT Entry	Adds a new OCT entry to the memory of the administration program
Delete OCT Entry	Deletes an OCT entry from the memory of the administration program
Modify OCT Entry	Modifies an OCT entry in the memory of the administration program
Display AT	Displays the elements of the AT
Add AT Entry	Adds a new AT entry to the memory of the administration program
Modify AT Entry	Modifies an AT entry in the memory of the administration program
Delete AT Entry	Deletes an AT entry from the memory of the administration program
Display SRT	Displays the elements of the SRT
Add SRT Entry	Adds a new SRT entry to the memory of the administration program
Delete SRT Entry	Deletes an SRT entry from the memory of the administration program
Modify SRT Entry	Modifies an SRT entry in the memory of the administration program
Store Schema	Transfers changes in the SRT, OCT, and AT from the memory of the administration program to the DSA or, in the case of errors, to a file

Function	Meaning
Load Schema	Transfers a schema that is stored in a file, as a result of an error, back into the memory of the administration program

Note that these functions are described in more detail in Chapter 9.

4.3.3 Shadow Administration

The Shadow Administration functions described in Table 4-3, which are supported by **gdsditadm**, allow the user to create and delete shadows of objects or subtrees in a DSA, to create and delete shadowing jobs, and to manage shadowing jobs. Shadowing jobs ensure that shadow objects are up-to-date by updating the shadows at regular intervals.

Table 4–3. Shadow Administration Functions

Function	Meaning
Create Shadows and Shadowing Job	Creates a shadowing job (on the local machine) and copies of the object or subtree in the target DSA or DSAs
Create Shadowing Job	Creates a new shadowing job (on the local machine)
Remove Shadows and Shadowing Job	Deletes a shadowing job and removes the copies in the target DSA
Remove Shadowing Job	Deletes a shadowing job
Update Shadowing Job	Activates or deactivates a shadowing job, or changes the update frequency of an active job
Display Shadowing Jobs	Displays existing shadowing jobs
Display Update Errors	Displays errors when updating objects/subtrees in the target DSA or DSAs
Remove Update Error	Cancels an update that failed on the target DSA so that it is not repeated at the next activation time of the update daemon process

These functions are described in more detail in Chapter 10.

4.3.4 Subtree Administration

The functions in Table 4-4, which are supported by **gdsditadm**, allow the user to append, move, copy, and delete subtrees, to change the **Master Knowledge** attribute of a subtree, and to change attribute values in a subtree.

Table 4–4. Subtree Administration Functions

Function	Meaning
Save Subtree	Writes the master or shadow information of a subtree to a file
Append Subtree	Appends a subtree that has been written beforehand to a file with the **Save Subtree** function, under a (new) parent node
Copy Subtree	Copies a subtree under a (new) parent node
Change Name / Move Subtree	Changes the name of an entry, which does not have to be an end node, or moves a subtree
Delete Subtree	Deletes a subtree in the master DSA or DSAs, or in the BIND DSA
Change Master	Changes the **Master Knowledge** attribute of all objects in a subtree
Modify Subtree	Changes the value of the attribute for all objects of a subtree with a specified attribute value

4.4 The gdscacheadm Command

The **gdscacheadm** command is used to invoke Cache Administration. Administrators can also invoke Cache Administration by selecting the **Logon to the DUA Cache** option from the Logon Menu Mask (Mask 1). Cache Administration supports object admininstration and cache update functions.

Table 4-5 shows the object administration functions that are supported by **gdsditadm** for logging into the DUA cache and that are supported by **gdscacheadm**.

Table 4–5. Object Administration Functions for Logging into the DUA Cache

Function	Meaning
Add Object	Adds a new object with attributes
Remove Object	Deletes an object
Display Objects	Displays entries in the DUA cache
Display Local and Default DSA	Displays the Distinguished Names of the local and default DSAs
Add Client Address	Informs the directory system of the PSAP address of the C-stub
Display Client Address	Displays the PSAP address of the C-stub
Delete Default DSA	Deletes a default DSA
Add Alias	Adds an alias for an object

The cache update functions display, activate, deactivate, and modify cache update jobs.

4.5 User Input

The administrator calls individual administration functions of the directory service by entering options in menu masks.

Table 4-6 gives an overview of the function keys available and their functionality. This is a model mapping; the function key mapping for your installation may differ.

Table 4–6. Function Keys and Functionalities

Key Name	Function Key	Functionality
F1	**F1**	Return from attribute masks to object list (Mask 18) in the case of **Display Objects** Go from object list (Mask 18) to object name mask (Mask 6) in the case of **Display Objects** Go from O/R-Name mask (Mask 34) to O/R-Address masks (Masks 28, 29, 30, 31, 32) in the case of **Modify Attribute** Go from O/R-Address (Mask 32) to PSAP mask (Mask 7a) in the case of **Modify Attribute**
<Return>	↵	Go to next field/line
<Menu>	**F2**	Execute function
<Scroll Up>	**F3**	Display previous page/object/entry
<Scroll Down>	**F4**	Display next page/object/entry
<Del Char>	**F5**	Delete a character
<Del Line>	**F6**	Delete a line
<Ins Char>	**F7**	Insert a character
F8	**F8**	Select a recurring attribute in Mask 6d
<Del>	**<Ctrl-c>**	Abort function and return
<End>	**<Ctrl-d>**	Terminate **gdssysadm**

Confirm input in mask fields by pressing **<Return>**. The following operations are possible in every mask input field (excluding toggle fields):

- Insert character at the cursor position by using **<Ins Char>**.
- Delete character at the cursor position by using **<Del Char>**. The character before the cursor position can be deleted by using **<Backspace>**.
- Delete the whole input field by using **<Del Line>**.

On toggle fields, select an option by pressing the space bar or by entering the option itself. Preset options are displayed in the mask.

In the last input field of a mask, confirm the input by pressing **<Return>** or **<Menu>**. The next mask in the function is then displayed or, if the function has ended, the calling menu mask is redisplayed.

Abort the current function in a mask by pressing **<Del>**. The calling menu mask is then displayed.

To page forward or back, press **<Scroll Down>** and **<Scroll Up>**. End the display by pressing **<Del>**. Use <↑> and <↓> to position the cursor on an element to be selected, and press **<Return>** to activate the selection.

Wildcards (*) in names are only permitted for retrieval functions.

Character entries of attribute values must end with an ' (apostrophe). Attribute values can, therefore, also contain blanks at the end. The attributes can also be input in hexadecimal; that is, in the form **x'...'**.

Note: Chapters 8, 9, 10, and 11 start by describing the masks used, along with their meaning and input options. This is followed by a description of each operation and its mask sequence.

4.6 Administration of GDS Using Input Files

The GDS commands, **gdsditadm** and **gdscacheadm**, accept ASCII input files that can automate specific administration procedures. Input files must provide mask entries in the same mask sequence as an administrator would provide interactively.

Every input in a mask field must be entered in a separate line in the input file. The end-of-line character is interpreted as a key input.

Any space remaining following a value in an input field must be filled with underscores so that longer values specified beforehand for the same field will be overwritten correctly.

Comments that begin with * (asterisk) and end with *: (asterisk, colon) are transferred from the input file to the output file. Comments must be enclosed between colons (:).

The following example of an input file adds some objects to the directory.

```
:*****TEST 1 (Add Object) DSA OP=4*1****:
:*** sccsid = @(#)t1.1addobj    7.2 91/06/24 (K Sys AP 11) ***:
```

```
:Directory ID:1
:Password:schmid
:Country:de
:Organization:Smith Ltd
:Organizational Unit:Sales
:Common name:Schmid
:Options:Logon to the Default DSA
:****Administration ****:
:Function:1
:****AddObject  US *******:
:Operation:1
:Object type number:2
:Country:US
:Object Class:C
:Auxiliary Object Class:NO
:Attribute name1:
:Attribute name2:
:Attribute name3:
:Attribute name4:
:Attribute name5:
:More:
:****AddObject  US/Smith Ltd *******:
:Operation:01
:Object type number:03
:Country:US
:Organization:Smith Ltd
:Object Class:Organization
:Auxiliary Object Class:NO
:Attribute name1:
:Attribute name2:
:Attribute name3:
:Attribute name4:
:Attribute name5:
:More:
:****AddObject  US/Smith Ltd/Sales *******:
:Operation:01
:Object type number:04
:country:US
:organization:Smith Ltd
:Organizational Unit:Sales
:Object Class:Organizational-Unit
```

```
:Auxiliary Object Class:NO
:Attribute name1:
:Attribute name2:
:Attribute name3:
:Attribute name4:
:Attribute name5:
:More:
:****AddObject  US/Smith Ltd/Sales/Huber *******:
:Operation:01
:Object type number:05
:Country:US
:Organization:Smith Ltd
:Organizational Unit:Sales
:User:Huber
:Object Class:Organizational-Person
:Auxiliary Object Class:NO
:Attribute name1:Surname
:Attribute name2:Telephone-Number
:Attribute name3:Telex-Number
:Attribute name4:Fax-Telephone-Number
:Attribute name5:
:More:
:Attribute name:Surname
:Attribute value:Huber'
:Attribute value:
:Attribute name:Telephone-Number
:Attribute value:12341234'
:Attribute value:
:Attribute name:
:Attribute value:
:Attribute value:
:Telex number:54377
:Country code:49
:Answerback:54
:FAX number:34445
:A3_Width:Y
:B4_Length:Y
:B4_Width:Y
:Fine resolution:Y
:Two dimensional:Y
:Uncompressed:Y
```

```
:Unlimited length:Y
:****AddObject  US/Smith Ltd/Sales/Sanjay,India *******:
:Operation:01
:Object type number:06
:Country:US
:Organization:Smith Ltd
:Organizational Unit:Sales
:User:Sanjay
:Org.-Unit-Name:India
:Object Class:Organizational-Person
:Auxiliary Object Class:NO
:Attribute name1:Surname
:Attribute name2:Telephone-Number
:Attribute name3:
:Attribute name4:
:Attribute name5:
:More:
:Attribute name:Surname
:Attribute value:jain'
:Attribute value:
:Attribute name:Telephone-Number
:Attribute value:1237261'
:Attribute value:
:Attribute name:
:Attribute value:
:Attribute value:
:****END****:
:Operation:00
:****END TEST****:
:Operation:00
```

The following is an example of an output file:

```
OUTPUT:
=======

BIND                    elapsed time:        0.0000 sec
:****Administration ****:
:****AddObject  US *******:
:****AddObject  US/Smith Ltd *******:
:****AddObject  US/Smith Ltd/Sales *******:
:****AddObject  US/Smith Ltd/Sales/Huber *******:
:****AddObject  US/Smith Ltd/Sales/Sanjay,India *******:
:****END****:
:****END TEST****:
```

If the command executes successfully, the return value is 0 (zero); otherwise, the value is nonzero.

Chapter 5

Installation and Day-to-Day Operation of GDS

This chapter describes how to install, start, stop, and monitor GDS.

5.1 Installation and Configuration Prerequisites

A minimum of 20 megabytes of memory is required for initial installation as well as additional disk space for the DIB. The administrator needs to define a globally unique cell name for each cell that accesses the directory service and to configure a Global Directory Agent (GDA) in each cell.

Each cell must meet the following requirements:

- Each cell must be configured as a CDS and Security client.
- The **rpcd** daemon must be running.
- The **/opt/dcelocal/dce_cf.db** file must be available on the machine and contain entries for the name of the cell and the name of the local host.
- Principal and account entries must be contained in the registry database for **hosts/***hostname***/gda**, and a **ktab** entry must be available for this principal.

- Each machine that runs as a server must have the following services and related processes (in parentheses) running:
  - — RPC (**rpcd**)
  - — Security (**secd**, **sec_clientd**)
  - — CDS (**cdsd**, **cdsadv**, **cdsclerk**)
  - — DTS (**dtsd**, **dts**_*device_name*_**provider**)
  - — GDA (**gdad**, **gda_child**)

5.2 Installing GDS

The procedure for installing GDS is described in detail in the *OSF DCE Administration Guide — Introduction*. The procedure is summarized as follows:

- Choose the **INSTALL** option from the DCE Main Menu.
- Choose the **GDS Server** option from the DCE Installation Menu. A list of GDS-related binary files that are being installed on the system is displayed.
- Choose the **Exit** option from the DCE Installation Menu.

5.3 Starting GDS

An administrator calls the **gdssysadm** process to administer the directory system. The menu mask shown in Figure 5-1 is then displayed.

Figure 5–1. Menu Mask (Part 1)

```
(diradm)                          DIRECTORY SYSTEM


  a - Administration of the directory information tree

  c - Configuration of a directory system

  b - Activation of a directory system installation

  d - Deactivation of a directory system installation

  s - Saving of local data to diskette/tape/file

  r - Restoring of saved data from diskette/tape/file

  f - Further functions
```

```
Your selection ! >
```

The GDS background processes are started by using the administration function **b**, which activates a directory installation.

5.4 Stopping GDS

The directory system is stopped with the administration function **d**, which deactivates a directory installation.

Before finishing, the background processes complete any activities that are currently running.

5.5 Monitoring GDS

In order to monitor most GDS processes, the trace system must be activated through the **l** option of the Menu Mask (Part 2) (see Figure 4-3). The trace system creates a set of log files for different processes. To switch on logging for the **gdssysadm** process, enter **X** in the Menu Mask (Part 1). Table 5-1 describes where these log files are located.

Table 5–1. Log File Subdirectories

Process	Subdirectory
Any application, **gdscacheadm**, **gdsditadm**	If the shell variable **D2_LOG_DIR** is set, the log files are created in the directory denoted by the value of **D2_LOG_DIR**; otherwise, they are created in **$HOME**
Cache	*dce_local*/**var/adm/directory/gds/cache**
C-stub	*dce_local*/**var/adm/directory/gds/cstub**
S-stub	*dce_local*/**var/directory/gds/adm/sstub**
DSA	*dce_local*/**var/directory/gds/adm/dir***x* where *x* represents the directory ID
gdssysadm	*dce_local*/**var/adm/directory/gds/adm**
Monitoring	*dce_local*/**var/adm/directory/gds/adm**

The names of the log files available for the processes are shown in Table 5-2. In the table, *pid* represents the process number.

Table 5–2. Log Files

Process	Log File
Any application, **gdscacheadm**, **gdsditadm**	**log_***pid***.l1** and **log_***pid***.l2** written successively (that is, when **l1** reaches its maximum size (2,000 records), logging continues in **l2**, and so on) **log.xds.pid** for XDS to CDSPI switch logging

Process	Log File
Cache	**cachetrace** and **cachetrace.SAV** written successively (that is, when the first reaches its maximum size (32 KB), **cachetrace** is copied to **cachetrace.SAV** and **cachetrace** is overwritten) **logfile** for fatal errors
C-stub	**log_***pid***.l1** and **log_***pid***.l2** written successively (that is, when **l1** reaches its maximum size (2000 records), logging continues in **l2**, and so on) **CMXLa***pid* and **CMXLb***pid* written successively for the transport system interface (that is, when **a** reaches its maximum size (256 KB), logging continues in **b**, and so on)
S-stub	**log_***pid***.l1** and **log_***pid***.l2** written successively (that is, when **l1** reaches its maximum size (2,000 records), logging continues in **l2**, and so on) **CMXLa***pid* and **CMXLb***pid* written successively for the transport system interface (that is when **a** reaches its maximum size (256 KB), logging continues in **b**, and so on)
DSA	**log_***pid***.l1** and **log_***pid***.l2** written successively (that is, when **l1** reaches its maximum size (2,000 records), logging continues in **l2**, and so on) **log_asn1.***pid* for the ASN.1 encoded messages
gdssysadm	**log_***pid*
Monitoring	**log_ipcchk**

The log files remain in the subdirectories after the processes end. If GDS is reactivated, the old log files are deleted and new log files are created. Some log files are in binary format.

The ASN.1 log files and transport interface log files are only written in conjunction with deactivation and activation of the directory system installation (see Sections 5.3 and 5.4). All other log files are written immediately.

Table 5-3 shows which tools must be used to evaluate the log files. All other log files are readable ASCII files.

Table 5–3. Tools for Evaluating Log Files

Log File	Command for Evaluating the Log File
log_*pid*.**l1**, **log_***pid*.**l2**	**gdsstep** *logfile*
CMXLa*pid*, **CMXLb***pid*	**gdscmxl -DXv** *logfile*

Logging is switched on immediately for the GDS system administration process (**gdssysadm**) when **X** is entered in the Menu Mask (Part 1) or Menu Mask (Part 2). When **X** is entered in either of these masks, logging for this process is switched off.

For all other processes, logging is switched on using the administration function **l** in the Menu Mask (Part 2), which activates the trace system, and is switched off again with the administration function **t**, which deactivates the trace system. Logging can be switched on and off when the system is both active and inactive.

If logging is to be switched on and off for any application process, the **gdsditadm**, or the **gdscacheadm** process, the shell variable **D2_LOG** must be set to **on** or **off** in the environment where the application is running.

5.5.1 Displaying Status with the gdsdirinfo Command

The **gdsdirinfo** command can be used to obtain useful information on all daemon processes running for GDS and on all current processes using GDS. The **gdsdirinfo** command reads all of the information from the GDS-specific shared memory area and writes to **stdout**. A 2-line header is printed first, followed by the information specific to the different processes (one line per process).

The following is an example of **gdsdirinfo** output:

```
# PROCTYPE PID DIRID IPCID STATE
#
   Monitor       4105       -          5          -
   DUA-Cache     4106       -          1          -
   C-Stub        4108       -          2          -
   S-Stub        4118       1          11         -
   S-Stub        4123       2          12         W1
   DSA           4130       1          31         -
   DSA           4125       2          32         -
   Dir-User      4300       -          31         R10
```

The following information is displayed:

PROCTYPE
The process type. The following types can occur: **Monitor**, **DUA-Cache**, **C-Stub**, **S-Stub**, **DSA**, and **Dir-User**.

PID The process identifier.

DIRID The directory identifier (1 - 20) with which the process is associated. If a process cannot be associated with a specific directory identifier (for example, the **DUA Cache** process), a dash is printed instead of a directory identifier number.

IPCID The IPC server ID with which the process is associated. This ID is used internally by GDS to establish an IPC association between an IPC client and an IPC server for sending distributed commands (for example, when activating and deactivating the trace system). The processes are assigned IPC server IDs as shown in Table 5-4.

Table 5–4. IPC Server IDs

ID Number	Process
1	**DUA Cache**
2	**C-stub**
5	**IPC-monitoring**
11-30	**S-stub**
31-50	**DSA** processes

If the process type is **Dir-User**, the IPC server ID displayed refers to the GDS (IPC) server (for example, **DUA-cache**, **C-stub**, **DSA**) with which this GDS client is associated. The relationship between IPCID and DIRID is important because it is easy to find the correct DSA or S-Stub if the directory ID is known. (The relationship is for S-Stub processes is Dir ID = IPCID - 10; the relationship for DSA processes is Dir ID = IPCID - 30).

STATE Describes the state of a GDS process during the startup phase or of a GDS client. Table 5-5 shows the valid states and their meanings.

If none of the specific states shown in Table 5-5 is associated with the process, a dash is printed.

Table 5-5 shows how a GDS process during the startup phase is designated by a state value of W*n*. GDS clients refer to any application. They are represented by **Dir-User** in **gdsdirinfo** output.

Table 5–5. GDS Process States

State Value	Process Type	Meaning
W1	C-Stub/S-Stub	C-Stub tries to read its own PSAP address from DUA cache
W3	DSA	DSA tries to read its own DSA name from the file
W4	DSA	DSA tries to read the internal schema
W5	DSA	DSA changes the schema object in the database
R1	GDS client	IPC association exists between GDS client and GDS server
R10	GDS client	DAP/DSP association exists between GDS client and GDS server

Note: A DSA process is also listed as a **Dir-User** if it chains a request through **S-Stub** to a remote DSA. **C-Stub** and **S-Stub** are also listed as **Dir-User** in the startup phase if they are trying to read the PSAP address from the DUA cache.

If the **gdsdirinfo** command is called when GDS is inactive, the following message is written to **stderr**:

```
gdsdirinfo: can't get IPC-resources (errno = 2)
```

If the command executes successfully, the exit value 0 (zero) is returned; otherwise, the value is 1 or 2.

(See the *OSF DCE Administration Reference* for a detailed description of the **gdsdirinfo** command.)

5.5.2 Interpreting Log Files with the gdsstep Program

Logging files that are generated by GDS applications can be subsequently evaluated and displayed with the **gdsstep** program. The logging records can be used by application programmers to debug their applications, and by users to determine network problems or protocol problems with remote systems.

(See the *OSF DCE Administration Reference* for a detailed description of the **gdsstep** program.)

Chapter 6

Initializing GDS

After GDS is installed, the administrator must perform the following steps:

1. Configure the directory system
2. Activate the directory system
3. Initialize the directory service

To perform these steps, the administrator calls **gdssysadm** by entering the following at the system prompt: **gdssysadm**. The menu mask shown in Figure 6-1 is then displayed.

Figure 6–1. Menu Mask (Part 1)

```
(diradm)                          DIRECTORY SYSTEM


  a - Administration of the directory information tree

  c - Configuration of a directory system

  b - Activation of a directory system installation

  d - Deactivation of a directory system installation

  s - Saving of local data to diskette/tape/file

  r - Restoring of saved data from diskette/tape/file

  f - Further functions
```

```
Your selection ! >
```

6.1 Configuring the Directory System

Once the directory system is installed, the system must be configured by using administration option **c** of the **gdssysadm** Menu Mask (Part 1). The menu mask shown in Figure 6-2 is then displayed.

Note: The directory system is the whole entity of the GDS installation; it consists of all the GDS processes. By introducing the concept of a directory ID, it is possible to generate DITs that have different schemas. This means that each directory ID represents its own "X.500 world." For example, directory ID 2 could represent an X.400 mail directory; whereas, directory ID 3 could represent a customer directory of a company. This concept is useful if the DITs are not combined into one DIT, which is possible in GDS.

Figure 6–2. Directory Service Configuration Mask

```
(confpar)                    DIRECTORY SYSTEM                    Configuration


 Which configuration mode ?: Creation of configuration data

For which directory ID should the operation be
performed ?: 2

Which configuration type ?: Client/Server system

How many clients have access to the directory system at the
same time ?: [1-256]: 16

How many server processes should be activated ? [1-256]: 2

Do you want to distribute 'update' information ?: NO
```

During configuration, the directory ID and the configuration type of the directory service are established. The configuration types are as follows:

- Client system
- Client/server system

It is possible to configure up to 20 IDs.

The Directory Service Configuration mask displays the following fields:

Which configuration mode ?

Select one of the following modes by pressing the space bar:

Creation of configuration data
To enter the configuration data for a new directory ID.

Deletion of configuration data
To delete the configuration data for a directory ID.

Display of configuration data
To display the configuration data for all the directory IDs that are configured (see Figure 6-3). Input in the remaining fields is not possible when this mode is selected.

Changing of configuration data
To change the configuration data for a directory ID.

For which directory ID should the operation be performed ?

Enter a directory ID in the range 1 to 20 for **Changing of configuration data** or **Display of configuration data**. Enter a directory ID in the range from 2 to 20 for **Creation of configuration data** or **Deletion of configuration data**. The required value can also be selected with the space bar.

The following entries are only displayed when **Creation of configuration data** or **Changing of configuration data** mode is selected:

Which configuration type ?

The configuration type of the directory service.

Select one of the following values by pressing the space bar:

- **Client system**
- **Client/Server system**

How many clients have access to the directory system at the same time ? [1-256]

The maximum number of clients possible is 256.

How many server processes should be activated ? [1-256]

The number of server processes to be activated depends on how many applications need to be able to access the DSA of the directory at the same time.

The number of server processes is determined by the system load. If the system load increases, additional server processes are activated automatically. If the system load decreases,

server processes are terminated automatically. However, the number of running server processes is never less than the value specified here.

Do you want to distribute 'update' information ?

Select one of the following values by pressing the space bar:

YES Distribute updates on master information to DSAs that have shadows

NO No distribution

When the mask for the configuration is filled, a configuration file is created (or updated). GDS starts the DSA processes as often as required by using this configuration file.

Notes: A new configuration does not become effective until the next time the directory installation is activated.

A configuration can only be deleted or changed if the directory installation is deactivated.

If the configuration is deleted or if the client/server system is reconfigured to a client system, all user data in the configuration is lost.

If the **Display of configuration data** option is selected in the **Which configuration mode ?** field, the mask in Figure 6-3 is displayed, giving details of all the directory IDs configured.

Figure 6–3. Mask for Displaying the Directory IDs Configured

```
(confpar)                        DIRECTORY SYSTEM                        Configuration


 Which Configuration mode ?: Display of configuration data

DIRECTORY-ID CONFIGURATION TYPE   SERVER-PROC  CLIENTS  UPDATE
    1          Clt/Srv-System         1          16      no
    3          Clt/Srv-System         1          16      no
```

To continue please press **<Return>**.

6.2 Activating the Directory System

After a directory system is configured, it must then be activated. To do this, select function **b** of the **gdssysadm** Menu Mask (Part 1) (Figure 6-1). This function activates a directory installation.

Starting background processes can take some time. You can display information on the processes by using the **i** function of Menu Mask (Part 2), which displays directory system status information. To display Menu Mask (Part 2), select function **f** from the Mask Menu (Part 1).

6.3 Initializing the Directory Service

The following sections describe the process for initializing the directory service in order to connect a new client or server to an already existing directory system.

6.3.1 Rules for Initializing the Directory Service

In order to make a newly configured directory system operable, some general guidelines must be observed regarding the following:

- DUA cache
- Schema object
- Master and shadow entries of DSAs
- First-level DSAs
- Objects mastered by superior DSAs

After the directory system is configured, the DUA cache does not contain any entries, and the local DSA only has the entry of a default schema.

6.3.1.1 DUA Cache

Each DUA cache must have an entry of the PSAP address of the client in order to set up a connection to a remote DSA. Each DUA cache also requires the entries of all DSAs, including their PSAP address, to which the DUA wishes to connect directly. Normally, the DUA cache has at least the name and the PSAP address of the local DSA.

When entering a DSA object in the DUA cache, the administrator can specify a special attribute called a **DSA-Type**, which can have the following values:

local DSA of the directory system that is in the same computer as the DUA. In order to access it, the local DUA does not need to establish a connection to the network. The local DSA can access other remote DSAs and can be accessed by remote DUAs and DSAs.

default A remote DSA that the administrator wishes to contact if the **Logon to the Default DSA** option is selected in the Menu Mask (Part 1). In order to access this DSA, the DUA needs to establish a connection to the network.

default/local Specifies the local DSA as a default DSA. For this purpose, the local DSA is entered in the DUA cache with **DSA-Type default/local'** instead of **DSA-Type local'**.

There is only one local or default/local DSA per directory ID. Several default DSAs can be entered in the DUA cache per directory ID. These can be either the local DSA or remote DSAs, or both. To set up a connection to the default DSA, the administrator must select the **Logon to the Default DSA** option in the Menu Mask (Part 1).

It is also possible to enter DSA objects in the DUA cache without specifying the attribute **DSA-Type**. To set up the connection to one of these remote DSAs, the administrator must select the **Logon to a Specific DSA** option in the Menu Mask (Part 1).

After the client address and DSA objects with **DSA-Type default/local'** or **local'** are entered, there can be a delay of up to one minute before the system starts working.

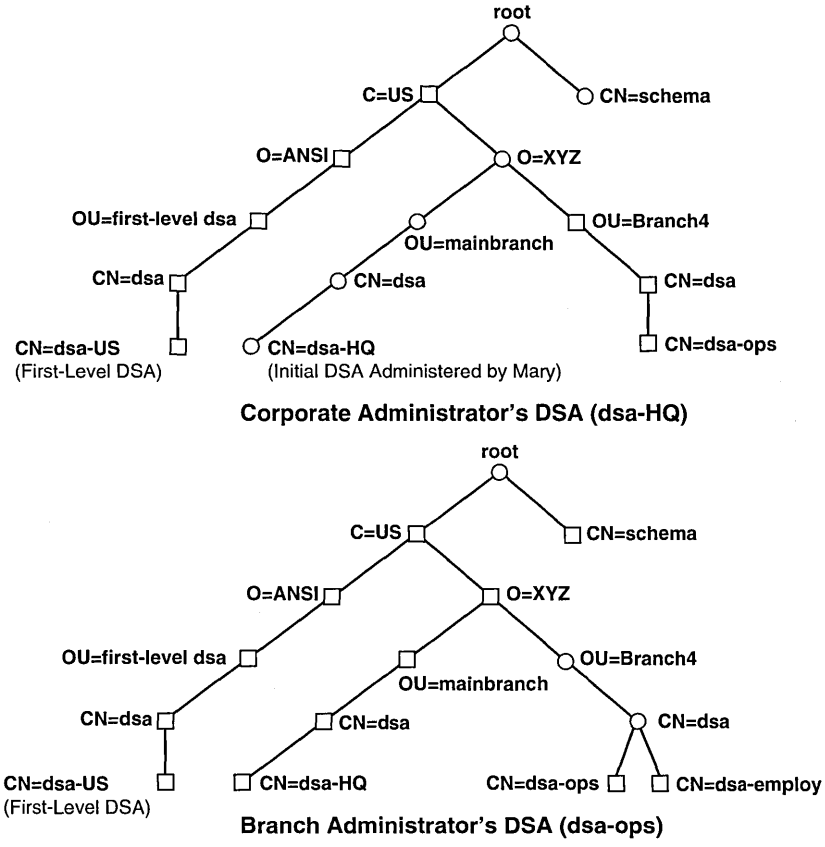
The values for **DSA-Type** that the branch network administrator entered on the sample Client worksheet and Client/Server worksheet are shown in Figures 3-6 and 3-7.

6.3.1.2 Schema Objects

Each DSA must contain the schema object under the root of the DIT. If the DSA has the master entry of the schema object, it is called an *initial DSA*. Other DSAs that contain a shadow of the schema object from an initial DSA constitute an administration domain.

For example, in Figure 6-2, one of the branch network administrator's DSAs, **dsa-ops**, contains a shadow of the schema object mastered by the corporate network administrator's DSA, **dsa-HQ**.

Figure 6–4. Schema Objects Under the Root of the DIT



Legend:

○ = Master Entry.

□ = Shadow Entry (with Master Knowledge information).

6.3.1.3 Master and Shadow Entries of DSAs

Each DSA must have a master entry of its own DSA and at least a shadow entry of one other DSA that knows the other DSAs in the directory.

For example, Figure 6-2 shows the entries for the DSAs in the branch network administrator's cell, **dsa-ops**, and the corporate network administrator's DSA, **dsa-HQ**. The **dsa-HQ** DSA has knowledge of all the DSAs in XYZ Corporation and the first-level DSA, **dsa-us** (giving **dsa-HQ** global knowledge of DSAs outside the US).

A GDS DSA can make a referral to another DSA only if a shadow of that DSA, including its PSAP address, is contained in the DIT. The same condition is also required for chaining.

Figure 6-2 shows that, because of the existence of the following objects, **dsa-ops** can send a referral (or perform chaining) to **dsa-HQ** and the first-level DSA **dsa-us**:

- **/C=US/O=XYZ/OU=mainbranch/CN=dsa/CN=dsa-HQ**
- **/C=US/O=ANSI/OU=first-level-dsa/CN=dsa/CN=dsa-US**

6.3.1.4 First-Level DSAs

A DSA that is master of an object under root (first-level object), except the schema object, is called a *first-level DSA*. Because this first level is not under the control of the directory system, it must be managed by a human directory administrator. The administrator must add shadow objects for all first-level objects and all other first-level DSAs, including the DSAs that master these objects.

The administrator of a first-level DSA must ensure that the names in the first level are unique and that only one DSA is master of an object.

6.3.1.5 Objects Mastered by Superior DSAs

Each DSA must have at least a shadow entry of the master DSA of the first-level object, which is superior to the local DSA object.

Note: It is advisable to add shadows of all DSAs with their PSAP addresses (these DSAs are masters of the superior nodes of the local DSA object). This also applies to superior nodes of objects that are mastered by the local DSA. This is necessary if these superior nodes are mastered by non-GDS DSAs.

6.3.2 Information Required for Initializing the Directory Service

The configuration worksheets described in Chapter 3 are designed to provide the client and client/server information that an administrator needs to initialize the directory service.

In summary, the following client information is required:

- Client address (that is, the PSAP address of the client stub on the local machine)
- Names and PSAP addresses of all default DSAs
- Names and PSAP addresses of all other remote DSAs that this DUA wants to contact directly (that is, not by following a referral received from another DSA)

The following client/server information is required:

- Client address (that is, the PSAP address of the client stub on the local machine)
- Name and the PSAP address of the local DSA
- Names and PSAP addresses of all default DSAs
- Names and PSAP addresses of all other remote DSAs that this DUA wants to contact directly (that is, not by following a referral received from another DSA)
- Indication of whether the default schema or other schema information is used.

6.3.3 Initialization Steps for the Directory Service

After the directory system is configured and activated, the system is still inaccessible. A number of initialization steps must be performed in sequence in order to set up the first directory system, or connect the directory system to one already running in a distributed environment.

There are several possible types of configurations of working directory systems:

- Initial client/server system
- Client system
- A client/server system with non-GDS DSAs
- A client/server system with DSAs that do not constitute an administrative domain
- An administrative domain that uses the default schema
- An administrative domain that does not use the default schema

There are 10 basic steps involved in initializing the directory system. The sequence in which these steps are performed varies according to the type of configuration. The following sections describe the sequence in which the initialization steps are to be performed for each configuration type. Section 6.3.4 provides a detailed description of each step.

6.3.3.1 Initial Client/Server System

Initializing a client/server system requires the following initialization steps. The step number in parentheses refers to the number of the step as listed in Section 6.3.4.

1. Enter the client address in the DUA cache (step 1).
2. Enter the local DSA, including its PSAP address, in the DUA cache (step 2).
3. Enter the local DSA as an object in the local DSA (step 8).

6.3.3.2 Client System

Initializing a client system requires the following initialization steps. The step number in parentheses refers to the number of the step as listed in Section 6.3.4.

1. Enter the client address in the DUA cache (step 1).
2. Enter all DSAs, including their PSAP addresses, that the client wants to connect to in the DUA cache (step 9).

6.3.3.3 Client/Server System with Non-GDS DSAs, or DSAs That Do Not Constitute an Administration Domain

Initializing a client/server system with non-GDS DSAs or DSAs that do not constitute an administration domain requires the following initialization steps. The step number in parentheses refers to the number of the step as listed in Section 6.3.4.

1. Enter the client address in the DUA cache (step 1).
2. Enter the local DSA, including its PSAP address, in the DUA cache (step 2).
3. Enter the local DSA as an object in the local DSA (step 8).
4. Enter all default DSAs, including their PSAP addresses, in the DUA cache (step 9).
5. Enter all other DSAs, including their PSAP addresses, that the client wishes to connect to in the DUA cache (step 9).
6. To speed up performance, copy other DSAs in the local DSA (cutting down on the amount of chaining and referral that is necessary to reach them) (step 7).
7. Enter a shadow of the local DSA in every other DSA that needs to refer to this DSA.
8. Enter the local DSA, including its PSAP address, in the DUA cache of all clients that need to be connected to this DSA.
9. If the local DSA is a first-level DSA, enter all first-level objects as shadows, including shadows of all other first-level DSAs with their PSAP addresses. Then enter a shadow of the local DSA in all other

first-level DSAs (for non-GDS implementations, knowledge of the existence of the new DSA has to be guaranteed by local means). The DSA objects in the DUA cache can be entered as default DSAs.

6.3.3.4 Client/Server System, Local DSA, and Initial DSA Constitute an Administration Domain and Use the Default Schema

Initializing a client/server system where the local DSA and initial DSA constitute an administration domain, and which uses the default schema requires the following initialization steps. The step number in parentheses refers to the number of the step as listed in Section 6.3.4.

1. Enter the client address in the DUA cache (step 1).
2. Enter the local DSA, including its PSAP address, in the DUA cache (step 2).
3. Enter the initial DSA, including its PSAP address, in the DUA cache (step 3).
4. Change the **Master Knowledge** attribute of the schema in the local DSA (step 4).
5. Enter a copy of the initial DSA in the local DSA (step 7).
6. Enter the local DSA as an object in the local DSA (step 10).
7. Enter all default DSAs, including their PSAP addresses, in the DUA cache (step 9).
8. Enter all other DSAs, including their PSAP addresses, that the client wishes to connect to in the DUA cache (step 9).
9. To speed up performance, copy other DSAs in the local DSA (cutting down on the amount of chaining and referral that is necessary to reach them) (step 7).
10. Enter a shadow of the local DSA in every other DSA that needs to refer to this DSA.
11. Enter the local DSA, including its PSAP address, in the DUA cache of all clients that need to be connected to this DSA.

12. If the local DSA is a first-level DSA, enter all first-level objects as shadows, including shadows of all other first-level DSAs with their PSAP addresses. A shadow of the local DSA must then be entered in all other first-level DSAs by the administrators of the other first-level DSAs. The DSA objects in the DUA cache can be entered as default DSAs.

6.3.3.5 Client/Server System, Local DSA, and Initial DSA Constitute an Administration Domain and Do Not Use the Default Schema

Initializing a client/server system where the local DSA and initial DSA constitute an administration domain, and which does not use the default schema requires the following initialization steps. The step number in parentheses refers to the number of the step as listed in Section 6.3.4.

1. Enter the client address in the DUA cache (step 1).
2. Enter the local DSA, including its PSAP address, in the DUA cache (step 2).
3. Enter the initial DSA, including its PSAP address, in the DUA cache (step 3).
4. Change the **Master Knowledge** attribute of the schema in the local DSA to be that of the initial DSA (step 4).
5. Copy the directory schema from the initial DSA to the local DSA (step 5).

   (The initial schema has changed. Therefore, it is necessary to copy over the modified schema to the local DSA. The local DSA also obtains a shadow of the initial DSA.)
6. Enter the local DSA, including its PSAP address, in the DUA cache using the new schema structure (step 6).
7. Enter the initial DSA, including its PSAP address, in the DUA cache using the new schema structure (as in step 6).
8. Enter the local DSA as an object in the local DSA (step 10).
9. Deactivate and activate GDS.

10. Enter all default DSAs, including their PSAP addresses, in the DUA cache (step 9).
11. Enter all other DSAs, including their PSAP addresses, that the client wishes to connect to in the DUA cache (step 9).
12. To speed up performance, copy other DSAs in the local DSA (cutting down on the amount of chaining and referral that is necessary to reach them) (step 7).
13. Enter a shadow of the local DSA in every other DSA that needs to refer to this DSA.
14. Enter the local DSA, including its PSAP address, in the DUA cache of all clients that need to be connected to this DSA.
15. If the local DSA is a first-level DSA, enter all first-level objects as shadows, including shadows of all other first-level DSAs with their PSAP addresses. A shadow of the local DSA must then be entered in all other first-level DSAs by the administrators of the other first-level DSAs. The DSA objects in the DUA cache can be entered as default DSAs.

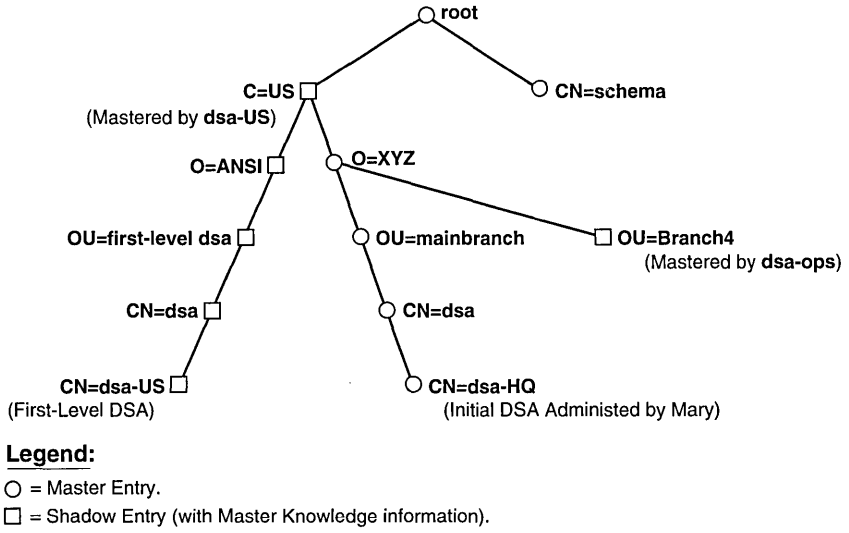
6.3.3.6 Sample Initialization of Client/Server System for an Administrative Domain Using the Default Schema

This section presents a sample initialization of a client/server system, **client/server1** by Branch Administrator 4.

This sample initialization demonstrates how Branch Administrator 4 adds a DSA to an administrative domain and creates the subtree of the entry **/C=US/O=XYZ/OU=Branch4** on **dsa-ops** so that it is mastered by **dsa-ops**.

Before starting the initialization of **dsa-ops**, Branch Administrator 4 tells Corporate Administrator 1 to create a shadow of **/C=US/O=XYZ/OU=Branch4)** with the **Master Knowledge** attribute equal to Branch Administrator 4's DSA (**/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**). No ACL will be set by Corporate Administrator 1. Figure 6-5 shows the tree on **dsa-HQ** after Corporate Administrator 1 adds the new shadow.

Figure 6–5. Shadow Entry on the Corporate Administrator's DSA (dsa-HQ)



Step 1: Enter the client address in the DUA cache

1. Branch Administrator 4 logs into the DUA cache.
2. Branch Administrator 4 enters the PSAP address of the client using Masks 3 through 7a in Object Administration.

   From the sample worksheet in Figure 3-6, the PSAP address is **TCP/IP!internet=192.35.18.1+port=21010**

Step 2: Enter the local DSA, including its PSAP address, in the DUA cache

1. Branch Administrator 4 enters the name and PSAP address of the local DSA in the DUA cache with the **Add Object** option in Object Administration Mask 4.
2. Branch Administrator 4 enters **7** as the structure rule of a DSA object in Mask 5.

3. Branch Administrator 4 enters the DN of the local DSA **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** and sets the **Auxiliary Object Class** field to **NO** in Mask 6.

4. Branch Administrator 4 enters the PSAP address of the local DSA, **TCP/IP!internet=192.35.18.1+port=21010** (the same one as the client address in step 1, but with a different T-Selector), in Masks 6d, 7, and 7a.

Step 3: Enter the initial DSA, including its PSAP address, in the DUA cache

1. Branch Administrator 4 enters the DN and PSAP address of the initial DSA, **dsa-HQ**, (Corporate Administrator 1's DSA) into the DUA cache.

Step 4: Change the Master Knowledge attribute of the schema in the local DSA

1. Branch Administrator 4 selects the **Logon to Default DSA** option in Mask 1.

2. Branch Administrator 4 selects option number 7 (**Modify Attribute**) in Mask 4.

3. Branch Administrator 4 enters 1 as the structure rule of the **Schema** object class.

4. Branch Administrator 4 changes the **Master Knowledge** attribute of the **Schema** object in Mask 8.

   Branch Administrator 4 needs to do this so that the local DSA no longer masters the schema object to allow the local DSA to be part of the administrative domain of **dsa-HQ**.

5. Branch Administrator 4 replaces the DN of the local DSA **dsa-ops (/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops)** with the DN of **dsa-HQ (/C=US/O=XYZ/ OU=mainbranch/CN=dsa/CN=dsa-HQ)**.

Step 7: Enter a copy of the initial DSA object in the local DSA

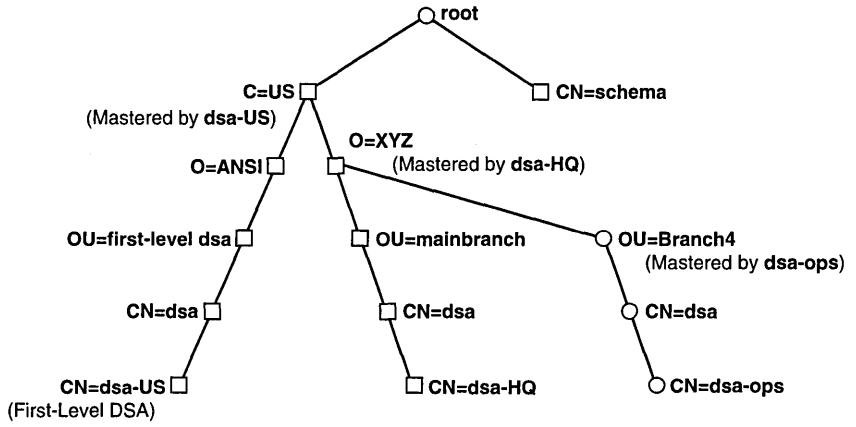
1. Branch Administrator 4 logs into the initial DSA by toggling the **Logon to a Specific DSA** option in Mask 1.

2. Branch Administrator 4 selects option number 4 (**Subtree Administration**) in Mask 3.
3. Branch Administrator 4 selects option number 3 (**Copy Subtree**) in Mask 16 to copy the subtree beneath the entry **/CN=US** from **dsa-HQ**.
4. Branch Administrator 4 enters 2 to specify the structure rule of Country in Mask 5.
5. Branch Administrator 4 enters the DN of the country and specifies **SINGLE OBJECT** for the **Object Interpretation** field.
6. Branch Administrator 4 toggles to **Specific DSA** as the source DSA **dsa-HQ** in Mask 17a by pressing the space bar.
7. Branch Administrator 4 enters the DN of the source (and initial) DSA in Mask 2.
8. Branch Administrator 4 defines the new parent node and specifies that the new entries will receive the ACL of the new parent, not the old ACL from **dsa-HQ** (which is set up for restrictive access by Corporate Administrator 1) in Masks 5 and 17b.

   Branch Administrator 4 also indicates that the existing entries are to be overwritten and specifies the target DSA (the bind DSA).

By copying the **Country** object (as part of the DN of the initial DSA), the **Copy Subtree** function automatically creates all the objects that are referenced in the **Master Knowledge** attribute of the **Country** object for **/CN=US**. The sequence of steps in step 7 are repeated so the subordinate entries below **Country** (such as **Organization** and **Organizational-Unit**) are copied until the tree shown in Figure 6-6 is created.

Figure 6–6. Master Entry on Branch Administrator 4's DSA (dsa-ops)



The first time step 7 is performed for **Country** (**/C=US**), the following objects are created:

- **/C=US**
- **/C=US/O=ANSI**
- **/C=US/O=ANSI/OU=first-level dsa**
- **/C=US/O=ANSI/OU=first-level dsa/CN=dsa**
- **/C=US/O=ANSI/OU=first-level dsa/CN=dsa/CN=dsa-US**

The second time step 7 is performed for **Organization** (**/C=US/O=XYZ**), the following objects will be created automatically:

- **/C=US/O=XYZ**
- **/C=US/O=XYZ/OU=mainbranch**
- **/C=US/O=XYZ/OU=mainbranch/CN=dsa**
- **/C=US/O=XYZ/OU=mainbranch/CN=dsa/CN=dsa-HQ**

Step 10: Enter local DSA as an object in the local DSA

1. Branch Administrator 4 creates **/C=US/ O=XYZ/ OU= Branch4.**
2. Branch Administrator 4 enters the local DSA, **dsa-ops**, as an object in the local DSA using the sequence of steps in Section 6.3.4. Using Masks 3 through 7a, as described in step 10 of Section 6.3.4, the branch network administrator adds the entry for **dsa-ops**.

Step 9: Enter the DSA, including its PSAP address that the client wishes to connect to in the DUA cache

1. Branch Administrator 4 enters all default DSAs (including their PSAP addresses) in the DUA cache by selecting the **Logon to the DUA Cache** option in Mask 1 and using Masks 3 through 7a.

   In Branch Administrator 4's sample Client/Server worksheet (Figure 3-7), Branch Administrator 4 has entered the names of the following as default DSAs: **dsa-ops**, **dsa-employ**, and **dsa-HQ**. Branch Administrator 4 can refer to the Client/Server worksheet and the GDS Remote and Non-GDS DSA worksheet (the filled-out sample worksheets shown in Figures 3-7 and 3-8) for the information on PSAP addresses.
2. Branch Administrator 4 enters all other DSAs (and PSAP addresses) in the DUA cache that the client wishes to connect to, using the procedures described in step 9 of Section 6.3.4.

Branch Administrator 4 also has the option of entering copies of other DSAs in the local DSA to speed up the referral mechanism.

Branch Administrator 4 should make sure that a shadow of **dsa-ops** is entered in every other DSA that needs to refer to **dsa-ops** and that **dsa-ops** (and its PSAP address) is entered in the DUA cache of all clients that need to be connected directly to **dsa-ops**.

6.3.4 Detailed Description (Mask Sequence) of Initialization Steps

This section describes the mask sequence of the initialization steps. When **Logon to the DUA Cache** is selected, the administration program tries to read the schema from the DSA (in order to generate masks in accordance with the actual schema information). The following message will appear, which you can ignore:

```
Schema from DSA cannot be read.
To continue press <CR>!
```

The administrator may need to log off from a DSA or DUA cache after a step is performed in order to be able to log into the DUA cache or local DSA. (For example, to initialize a client/server system the administrator must perform steps 2 and 8 in sequence. Step 2 requires a login to the DUA cache and step 8 requires a login to the local DSA. After step 2 is performed, the administrator must log out of the DUA cache and then log into the local DSA.)

This means that the administrator must return to Mask 1 and then log into the required DSA or DUA cache. The administrator always logs in anonymously; that is, there is no input for the **Password** or **DN** fields in Mask 1.

(See Chapters 7 and 8 for detailed descriptions of the masks.)

Step 1: Enter the client address in the DUA cache

Log into the DUA cache from Mask 1 by selecting the **Logon to the DUA Cache** option, then enter the directory ID of the configuration to be initialized. Proceed through the masks as follows:

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 5 (**Add Client Address**).

Mask 7a: Enter the presentation address of the client system.

Note: The C-stub requires up to one minute to obtain its client address from the DUA cache. The DUA cannot access remote DSAs until then.

Step 2: Enter the local DSA, including its PSAP address, in the DUA cache.

Log into the DUA cache by selecting the **Logon to the DUA Cache** option and by entering the directory ID of the configuration to be initialized in Mask 1.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 1 (**Add Object**).

Mask 5: Enter the structure rule of a DSA object. In the default schema, this is **Common-Name** (7). The mask displays the default schema structure.

Mask 6: Enter the DN of the local DSA in accordance with the default schema. The **Structural Object Class** field must be set to **Directory-Service-Agent**. The auxiliary object class field must be set to **NO**.

The DN of a DSA can only be entered in accordance with the default schema. If the DN does not correspond to the schema on the initial DSA, enter a temporary DN, then enter the correct DN later.

If the initial DSA is being initialized, the definitive DN can be entered immediately.

Mask 6d: The **Presentation-Address** attribute is automatically selected. Scroll to **DSA-Type**.

Mask 7: Enter the attribute name **DSA-Type** and the attribute value **local'** or **default/local'**.

Mask 7a: Enter the presentation address of the local DSA.

Note: The DSA requires up to one minute to obtain its name and PSAP address from the DUA cache. The DSA is not accessible until then.

Step 3: Enter the initial DSA, including its PSAP address, in the DUA cache.

Log into the DUA cache by selecting the **Logon to the DUA Cache** option and by entering the directory ID of the configuration to be initialized in Mask 1.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 1 (**Add Object**).

Mask 5: Enter the structure rule of a DSA object. In the default schema, this is **Common-Name** (7). The mask displays the default schema structure.

Mask 6: Enter the DN of the initial DSA in accordance with the default schema.

Set the **Structural Object Class** field to **Directory-Service-Agent** by pressing the space bar.

Set the **Auxiliary Object Class** field to **NO** by pressing the space bar.

The DN of a DSA can only be entered in accordance with the default schema. If the DN does not correspond to the schema on the initial DSA, a temporary DN can be entered. The correct DN is then entered later.

Mask 6d: The **Presentation-Address** attribute is automatically selected. Leave this mask by using **<Menu>** without selecting any other attribute.

Mask 7a: Enter the presentation address of the initial DSA.

Step 4: Change the Master Knowledge attribute of the schema in the local DSA.

Log into the local DSA by selecting the **Logon to the Default DSA** option and by entering the directory ID of the configuration to be initialized in Mask 1.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 7 (**Modify Attribute**).

Mask 5: Select the structure rule **Common-Name** (1).

Mask 6: Select the common name of the schema object.

Set the **Structural Object Class** field to **Schema** by pressing the space bar.

Set the **Auxiliary Object Class** field to **NO** by pressing the space bar.

Mask 6d: Select the **Master Knowledge** attribute.

Mask 8: The **Master Knowledge** attribute of the schema is displayed in the **Old Value** and **New Value** fields.

Enter the new value for **Master Knowledge** attribute.

Step 5: Copy the directory schema from the initial DSA to the local DSA.

Log into the initial DSA by selecting the **Logon to a Specific DSA** option and by entering the directory ID of the configuration to be initialized in Mask 1.

Mask 3: Select option number 4 (**Subtree Administration**).

Mask 16: Select option 3 (**Copy Subtree**).

Mask 5: Select the structure rule of the Schema, **Common Name** (1).

Mask 6: Select the common name of the schema object.

Set the **Object Interpretation** field to **SINGLE OBJECT** by pressing the space bar.

Mask 17a: Set the **Source DSA** field to **BIND DSA** by pressing the space bar.

Mask 5: Define the new parent node by entering option number 00 (**ROOT**) as the structure rule.

Mask 17b: Define **Target DSA** and other parameters for **Copy Subtree** by selecting the following values by pressing the space bar:

Overwrite existing entries:	**YES**
New entries protected by:	**ACL of the new parent**
Target DSA:	**SPECIFIC DSA**

Mask 2: Enter the DN of local DSA as entered in the DUA cache.

Step 6: Enter the local DSA, with its PSAP address, in the DUA cache using the new schema structure.

Log into the DUA cache by selecting the **Logon to the DUA Cache** option and by entering the directory ID of the configuration to be initialized in Mask 1.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 1 (**Add Object**).

Mask 5: Enter the structure rule of a DSA object. In the default schema this is **Common-Name** (7). The mask displays the structure of the schema that is taken from the initial DSA.

Mask 6: Enter the the DN of the local DSA in accordance with the schema taken from the initial DSA.

The structural object class of the DSA must be set to **Directory-Service-Agent**. The **Auxiliary Object Class** field must be set to **NO**.

Enter the definitive DN of the local DSA.

Mask 6d: The **Presentation-Address** attribute is automatically displayed. Select **DSA-Type**.

Mask 7: Enter the attribute name **DSA-Type** and the attribute value **local'**or **default/local'**.

Mask 7a: Enter the presentation address of the local DSA.

To enter other DSAs that the DUA wishes to connect to directly in the DUA cache, repeat the mask sequence from Mask 4 to Mask 7a; these other DSAs can then be declared as **default'**.

Step 7: Enter a copy of the initial DSA object in the local DSA.

Log into the local DSA by selecting the **Logon to the Default DSA** option or the **Logon to a Specific DSA** option (if the local DSA is not the default DSA) from Mask 1, then enter the directory ID of the configuration to be initialized.

Mask 3: Select option number 4 (**Subtree Administration**).

Mask 16: Select option number 3 (**Copy Subtree**).

Mask 5: Define the object to be copied by entering the structure rule of **Country** (2 in the default schema).

Mask 6: Enter the DN of the country (mastered by the initial DSA).

Set the **Object Interpretation** field to **SINGLE OBJECT** by pressing the space bar.

Mask 17a: Set the **Source DSA** field to **SPECIFIC DSA** by pressing the space bar.

Mask 2: Enter the DN of the initial DSA.

Mask 5: Define new parent node by entering the structure rule of parent node as **00 ROOT**.

Mask 17b: Define **Target DSA** and other parameters for **Copy Subtree** by selecting the following values by pressing the space bar:

Overwrite existing entries:	**YES**
New entries protected by:	**ACL of the new parent**
Target DSA:	**BIND DSA**

To speed up performance, it is more efficient to copy other DSAs in the local DSA (cutting down on the amount of chaining and referral that is necessary to reach them) in addition to the initial DSA.

When copying the **Country** object (that is part of the distinguished name of the initial DSA), the **Copy Subtree** function will automatically create all the objects that are referenced in the **Master Knowledge** attribute of that object (which is the first-level DSA that is then automatically created).

Masks 3 through 17b must then be called again for the next object (**Organization**), and so on, until the initial DSA gets automatically created.

Step 8: Enter the local DSA as an object in the local DSA.

Log into the local DSA from Mask 1 by selecting the **Logon to the Default DSA** option or the **Logon to a Specific DSA** option (if the local DSA is not the default DSA) from Mask 1, and then enter the directory ID of the configuration to be initialized.

If the higher-level nodes of the entry are missing, they must first be specified with the mask sequence from Mask 3 to Mask 6.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 1 (**Add Object**).

Mask 5: Enter the structure rule of a DSA object. In the default schema, this is **Common-Name** (7).

Mask 6: Enter the DN of the local DSA. The **Structural Object Class** field must be set to **Directory-Service-Agent**. The **Auxiliary Object Class** field must be set to **NO**.

Mask 6d: The **Presentation-Address** attribute is automatically displayed. Select the **User-Password** attribute.

Mask 7: Enter the attribute name **User-Password** and enter any password for the attribute value. The password must end with ' (apostrophe).

Mask 7a: Enter the presentation address of the local DSA.

If the initial DSA does not automatically receive a copy of the master entry of the local DSA as a mandatory shadow entry (see Section 8.2.1), the administrator must ensure that the initial DSA receives a shadow entry of the local DSA.

To protect the newly added master entries in the DSA against write access by unauthorized users, the administrator must first add his or her DN as an object to a node for whose naming attribute he or she has write access to. The administrator must then specify the access rights for the newly added nodes.

Step 9: Enter the DSA, including its PSAP address, that the client wishes to connect to in the DUA cache.

Log into the DUA cache from Mask 1 by selecting the **Logon to the DUA Cache** option, then enter the directory ID of the configuration to be initialized.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 1 (**Add Object**).

Mask 5: Enter the structure rule of a DSA (7 in the default schema). The mask displays the default schema structure.

Mask 6: Enter the DN that the client wishes to connect to, in accordance with the default schema.

The **Structural Object Class** field must be set to **Directory-Service-Agent**. The **Auxiliary Object Class** field must be set to **NO**.

The DN of a DSA can only be entered in accordance with the default schema. If the DN does not correspond to the schema of the initial DSA, enter a temporary DN, then enter the correct DN later.

If the initial DSA is initialized, the definitive DN can be entered immediately.

Mask 6d: The **Presentation-Address** attribute is automatically displayed. Select **DSA-Type** if this DSA is a default DSA. If **DSA-Type** is selected, then Mask 7 is shown.

Mask 7: Enter the attribute name **DSA-Type** and the attribute value **default'**.

Mask 7a: Enter the presentation address of the local DSA.

Step 10: Enter local DSA as an object in the local DSA.

Log into the local DSA by selecting the **Logon to the Default DSA** option or the **Logon to a Specific DSA** option (if the local DSA is not the default DSA), then enter the directory ID of the configuration to be initialized.

Mask 3: Select option number 4 (**Subtree Administration**).

Mask 16: Select option number 3 (**Copy Subtree**).

Mask 5: Enter the structure rule of the object (this is the object for whose subtree the local DSA will be responsible).

Mask 6: Enter the DN of the object to be copied. Set the **Object Interpretation** field to **SINGLE OBJECT** by pressing the space bar.

Mask 17a: Specify **SPECIFIC DSA** in the **Source DSA** field.

Mask 2: Enter the DN of the initial DSA.

Mask 5: Enter the structure rule of the parent node.

Mask 17b: Define **Target DSA** and other parameters for **Copy Subtree** by selecting the following values by pressing the space bar:

Overwrite existing entries:	**YES**
New entries protected by:	**ACL of the new parent**
Target DSA:	**BIND DSA**

Mask 16: Enter option number 0 (Exit).

The following sequence creates all the superior nodes of the local DSA object by calling the mask sequence several times. For all intermediate nodes, no attributes are assigned.

Mask 3: Select option number 1 (**Object Administration**).

Mask 4: Select option number 1 (**Add Object**).

Mask 5: Enter the structure rule of a DSA (7 in the default schema).

Mask 6: Enter the DN of the local DSA. The **Structural Object Class** field of the DSA must be set to **Directory-Service-Agent**. The **Auxiliary Object Class** field must be set to **NO**.

Mask 6d: The presentation address attribute is automatically displayed. Select the **User-Password** attribute.

Mask 7: Enter the attribute name **User-Password** and specify any password for the attribute value. The attribute value must end with ' (apostrophe).

Mask 7a: Enter the presentation address of the local DSA.

To protect the newly added master entries in the DSA against write access by unauthorized users, the administrator must first add his or her DN as an object to a node for whose naming attribute he or she has write access. The administrator must then specify the access rights for the newly added nodes.

The objects that the administrator has created on the local DSA using the **Copy Subtree** function are not full shadows of the master entries, because **Copy Subtree** is subject to the ACL.

Chapter 7

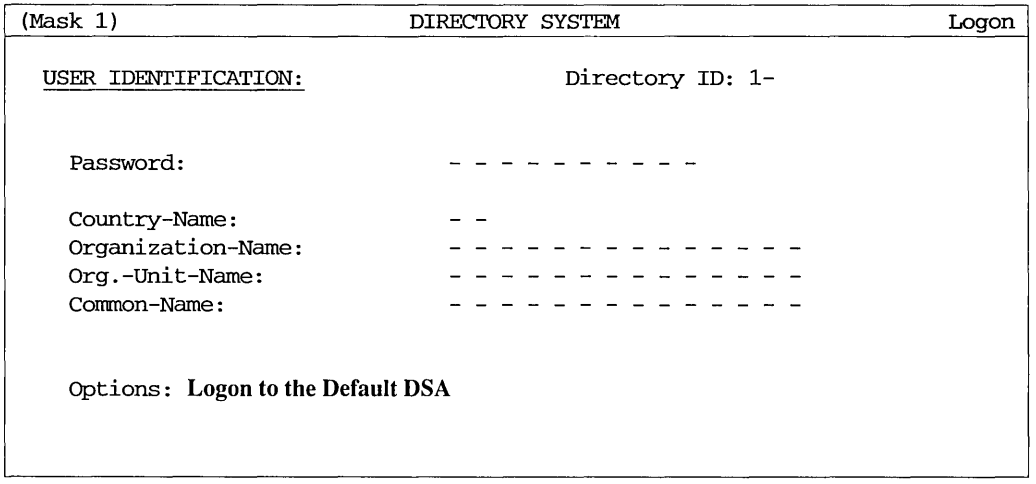
Logging into a DSA or the DUA Cache

To log into a DSA or the DUA cache, select the **Administration of the directory information tree** option from the Menu Mask (Part 1) in **gdssysadm** (see Figure 4-2). The masks used to administer the directory system are then displayed. Alternatively, you can call the **gdsditadm** or **gdscacheadm** process directly.

7.1 Mask 1: User Identification

When the **Administration of the directory information tree** option is selected in the Menu Mask (Part 1), Mask 1 (shown in Figure 7-1) is displayed. This mask allows the administration functions of the directory system to be accessed.

Figure 7–1. Mask 1: User Identification



When the cache administration process (**gdscacheadm**) is called, it is only possible to log into the cache. In this case, only the field for entering the **Directory ID** and the **Options** fields are displayed.

Mask 1 displays the following fields:

Directory ID

Enter the directory ID of the directory to be administered.

Password

Enter your password (password of the administrator object).

An anonymous user (that is, a user with no DN and no password) is always accepted. In this case, all Public attributes and attributes that are not protected by an ACL can be read and modified.

Country-Name
Organization-Name
Org.-Unit-Name
Common-Name

Enter the name parts of the DN of the administrator object.

Options

Select one of the following options by pressing the space bar:

- **Logon to the Default DSA** (default)
- **Logon to a Specific DSA** (see Section 7.2)
- **Logon to the DUA Cache** (see Section 7.4)
- **Changing Name Structure**

If **Changing Name Structure** is selected, all structure rules are displayed in Mask 5 (see Figure 8-4 in Chapter 8). Only structure rules that represent a person can be selected. The only structure in the default schema that represents a person is rule number 5.

After you make the appropriate selections, Mask 1 shows the selected name structure for entering your DN. This structure rule is also used for the next login to a DSA.

A number of default DSAs are possible. If the **Logon to the Default DSA** option is selected, then the administration program logs into the first available default DSA. If no default DSA is available in the cache, or none of the default DSAs are available, then the administration program attempts to log into the local DSA.

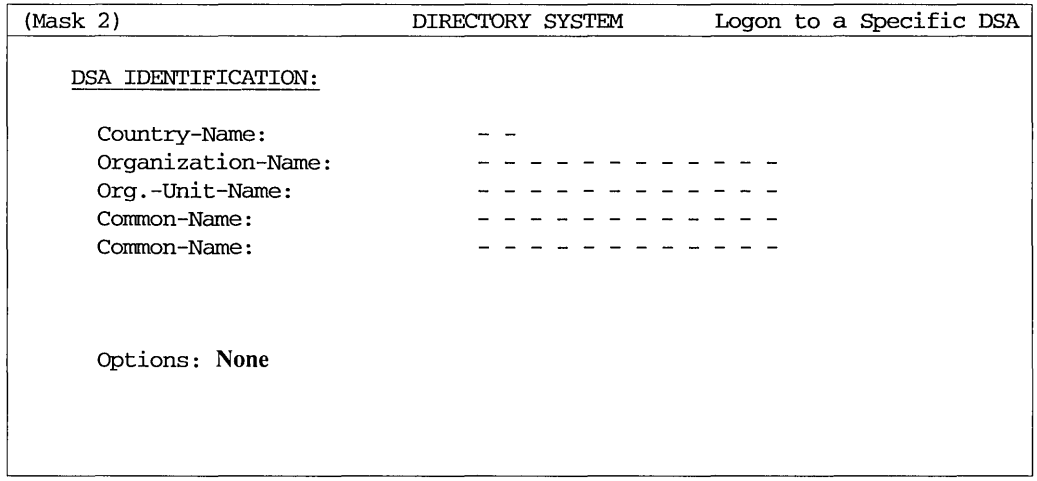
When the cache administration process (**gdscacheadm**) is called, the **Logon to the DUA Cache** option is displayed and no other option can be selected.

Note: The anonymous user can read and write all attributes, provided no DN is entered in the schema ACL after the directory service is configured.

7.2 Mask 2: DSA Identification

To connect to a specific DSA (using the **Logon to a Specific DSA** option), the administrator needs to enter the DN in Mask 2 (Figure 7-2).

Figure 7–2. Mask 2: DSA Identification



Mask 2 displays the following fields:

Country-Name

Organization-Name

Org.-Unit-Name

Common-Name

Common-Name

Enter the DN of the DSA to be selected.

The following options can be selected by pressing the space bar:

- **None** (default)
- **Changing Name Structure**

If the **Changing Name Structure** option is selected, all structure rules are displayed in Mask 5. Only structure rules representing a DSA can be selected.

The only structure in the default schema that represents a DSA is structure rule 7.

After selection, Mask 2 (see Figure 7-2) displays the selected name structure for entering the DN. This structure rule is also used for the next login.

7.3 Mask 3: Administration Functions

When the administrator successfully logs into a DSA, Mask 3 is displayed (Figure 7-3). This mask is used to select the required administration functions.

Figure 7–3. Mask 3: Administration Functions

```
(Mask 3)                        DIRECTORY SYSTEM                    Administration

  ADMINISTRATION FUNCTIONS

    0  Exit

    1  Object Administration

    2  Schema Administration

    3  Shadow Administration

    4  Subtree Administration

Current DSA ....................................................
            ....................................................

            Which function ? -
```

Mask 3 displays the following field:

Which function ?

Enter the number of the selected administration function (0 to 4).

If the **Schema Administration** option is selected in Mask 3, the SRT, OCT, and AT are loaded automatically in the administration program memory.

7.4 Logging into the DUA Cache

To control the information stored in the cache, log into the local machine using the **Logon to the DUA Cache** option in Mask 1. When this option is selected, a different version of Mask 3 is displayed, as shown in Figure 7-4.

Figure 7–4. Mask 3: Administration Functions Under the DUA Cache

```
(Mask 3)                    DIRECTORY SYSTEM                    Administration

ADMINISTRATION FUNCTIONS:

  0  Exit

  1  Object Administration

  2  Cache Update

Which function ? -
```

Mask 3 displays the following field:

Which function ?

Enter the number of the selected administration function (0 to 2).

The **Object Administration** option provides the following functions for managing objects and information in the cache:

- **Add Object**
- **Remove Object**
- **Display Objects**

- **Display Local and Default DSA**
- **Add Client Address**
- **Display Client Address**
- **Delete Default DSA**
- **Add Alias**

An administrator generally defines one or more DSAs to be contacted. If the administrator wants to log into the default DSA, the DUA connects internally to the first accessible default DSA. The administrator can use the **Delete Default DSA** function to contact different DSAs (for example, if the DSA network grows, or if a DSA is substituted by another DSA). Alternatively, the administrator can delete a name that is entered incorrectly.

The client address is required to set up a distributed system so that access to remote DSAs is supported. The administrator can add or display the client address by using the **Add Client Address** and **Display Client Address** functions.

The administrator can use the **Display Objects**, **Add Object**, **Remove Object**, and **Add Alias** functions to determine the current objects resident in the cache, add an object that requires frequent access by users, remove objects that are accessed infrequently, and create alias entries.

Selecting the **Cache Update** option from Mask 3 enables the administrator to modify the update frequency of all the information stored in the cache, activate an inactive **Cache Update** job, or deactivate an active **Cache Update** job to modify the update frequency.

The administrator selects one of the three update frequency values: **HIGH**, **MEDIUM**, or **LOW**.

If several objects change over a very short period of time in the DSA and the cache needs to be changed as soon as possible, the administrator should choose high update frequency. A low update frequency is chosen if there are few changes in the DSA over a short period of time.

7.5 XDS API Function Calls and the DUA Cache

Application programs can make directory service calls using XDS API that access information in the DUA cache. Programmers developing XDS API application programs may need to coordinate with the administrator to make sure that the appropriate information is available in the DUA cache.

XDS API calls that access the DUA cache are passed on to the DUA library. The DUA first looks in the DUA cache (if requested by the user) to see if the requested information is already available on the local machine. If it is not available, the DUA queries a DSA. If the DSA has the requested information, it returns the results to the DUA. If the DSA does not have this information, the query can proceed by means of chaining or a referral. In either case, different DSAs are queried until the information is found. It is cached (if requested by the user) in the DUA cache and the results are returned to the application program.

Chapter 8

Object Administration

The object administration functions are called when option 1 (**Object Administration**) is selected in Mask 3 (see Section 7.3).

This chapter describes the masks required to administer the objects in a DSA along with their fields and input options. The mask descriptions are followed by a description of each individual operation, its function, and its mask sequence.

8.1 Masks

Each of the following sections describes one of the masks used for object administration. (See Section 4.4 for more information on how to use masks and the conventions used for describing masks in this document.)

8.1.1 Mask 4: Object Operations

Mask 4 (Figure 8-1) displays the operations you can peform after logging into a DSA.

Figure 8–1. Mask 4: Object Operations After Logging into the DSA

```
(Mask 4)                          DIRECTORY SYSTEM          Object Administration


  OPERATIONS                  Entry Type: MASTER


   0  Exit
   1  Add Object
   2  Remove Object
   3  Display Objects (Global Master Info)
   4  Display Objects (Entries in Current DSA)
   5  Add Attributes
   6  Delete Attributes
   7  Modify Attribute
   8  Add Alias
   9  Modify RDN
```

```
Which operation ?
```

Mask 4 displays the following fields:

Entry Type This field defines the type of object entries to be processed. Select one of the following values by pressing the space bar:

MASTER To maintain the master entries of the objects (default).

SHADOW To maintain the shadow entries of the objects. This entry is not relevant for operations 3 and 4. (However, it is recommended that you use Shadow Administration to administer shadows when working with other GDS DSAs. When working with non-GDS DSAs, this is the only way shadows can be administered.)

Which operation?

Enter the number of the selected object administration operation (0 to 9).

Mask 4 (Figure 8-2) displays the operations you can perform after logging into the DUA cache.

Figure 8–2. Mask 4: Object Operations After Logging into the DUA Cache

```
(Mask 4)                        DIRECTORY SYSTEM          Object Administration

    OPERATIONS

       0  Exit
       1  Add Object
       2  Remove Object
       3  Display Objects
       4  Display Local and Default DSA
       5  Add Client Address
       6  Display Client Address
       7  Delete Default DSA
       8  Add Alias




    Which operation ?
```

Mask 4 displays the following field:

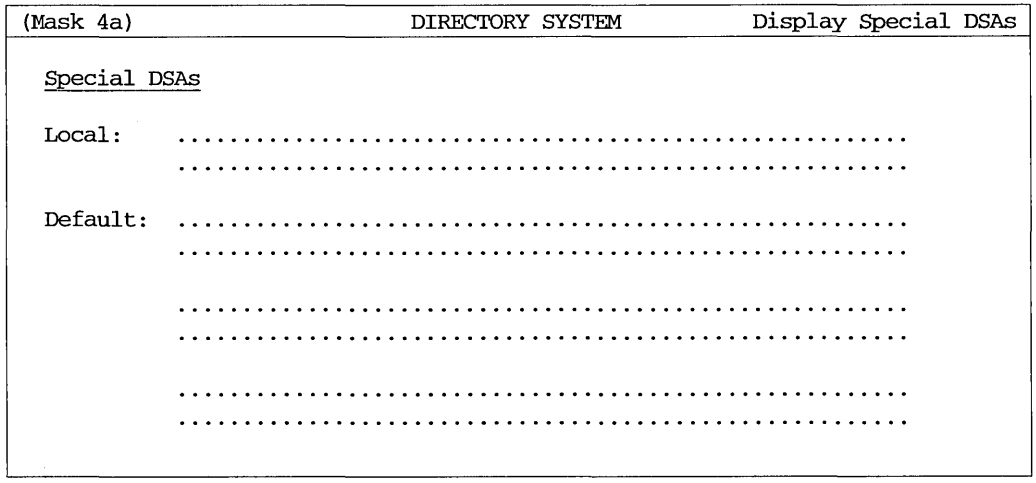
Which operation?

Enter the number of the selected object administration operation (0 to 8).

8.1.2 Mask 4a: Special DSAs

Mask 4a (Figure 8-3) displays the names of the local DSA and the default DSAs, as they are stored in the DUA cache.

Figure 8–3. Mask 4a: Special DSAs



If the default DSA is also the local DSA, the DN shown in the mask begins with **1:**. Otherwise, it begins with **0:**. (It is a remote DSA.)

If the DUA cache contains more than four default DSAs, **<Return>** or **<Menu>** can be used to page down and display other default DSAs.

8.1.3 Mask 5: Structure Rule

Mask 5 (Figure 8-4) is used to select the structure rule to be processed. It displays the name and name structure of every structure rule entered in the SRT. If the SRT contains more than 12 structure rules, **<Scroll Up>** and **<Scroll Down>** can be used to page through all the structure rules.

Figure 8–4. Mask 5: Structure Rule

```
(Mask 5)                    DIRECTORY SYSTEM                    operation

Structure Rule                          Name Structure

01 ..........................    ..............................
02 ..........................    ..............................
03 ..........................    ..............................
04 ..........................    ..............................
05 ..........................    ..............................
06 ..........................    ..............................
07 ..........................    ..............................
08 ..........................    ..............................
09 ..........................    ..............................
10 ..........................    ..............................
11 ..........................    ..............................
12 ..........................    ..............................
```

```
Which Structure Rule ?
```

In Mask 5, *operation* will be one of the following depending on the operation being performed:

Logon to a Specific DSA
Add Object
Remove Object
Display Objects
Add Attributes
Delete Attributes
Modify Attribute
Add Alias
Modify RDN
Delete Default DSA

Mask 5 displays the following field:

Which Structure Rule ?
Enter the number of the structure rule to be processed.

Figure 8-5 shows name structures displayed in Mask 5.

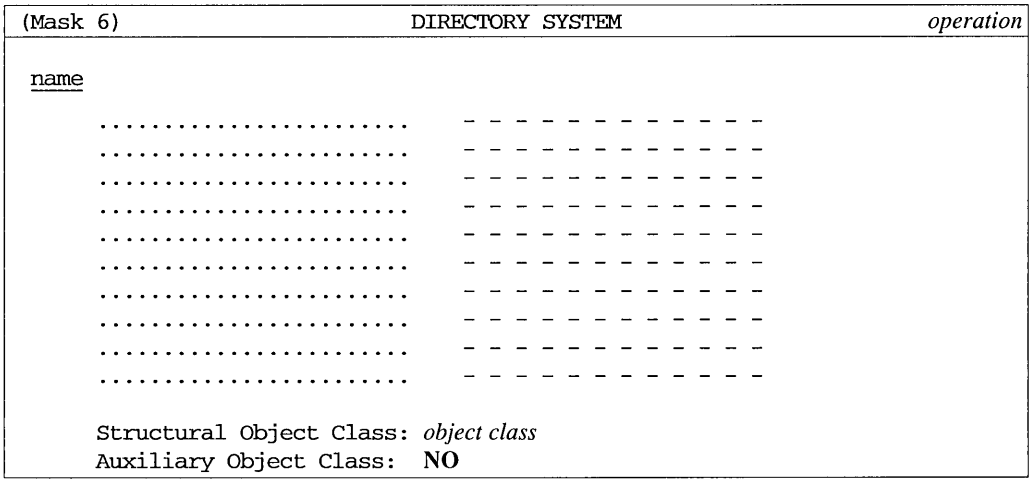
Figure 8–5. Mask 5: Sample Structure Rules

```
(Mask 5)                        DIRECTORY SYSTEM                        operation

Structure Rule                          Name Structure

 01 Common-Name                          01
 02 Country-Name                         02
 03 Organization-Name                    02-03
 04 Org.-Unit-Name                       02-03-04
 05 Common-Name                          02-03-04-05
 06 Common-Name                          02-03-04-06
    Org.-Unit-Name
 07 Common-Name                          02-03-04-05-07
 08 Locality-Name                        02-08
 09 Common-Name                          02-08-09
 10 Common-Name                          02-08-10
    Street-Address

Which Structure Rule ?
```

8.1.4 Mask 6: Object Name

Mask 6 (Figure 8-6) is used to display or enter the object name. The names of the naming attributes (stored in the AT) are displayed for the structure rule selected in Mask 5.

Figure 8–6. Mask 6: Object Name



In Mask 6, *operation* will be one of the following depending on the operation to be performed:

Add Object
Remove Object
Display Objects
Add Attributes
Delete Attributes
Modify Attribute
Add Alias
Modify RDN
Delete Default DSA

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **name** field can contain any of the following fields:

Object Name
Alias Name
Aliased Name
New Object Name

Mask 6 displays the following fields:

Structural Object Class:
This field is generated dynamically according to the selected structure rule (it is only displayed if *operation* is **Add Object**, **Display Objects**, **Add Attributes**, **Delete Attributes**, or **Modify Attribute**). If the operation is **Display Objects**, a wildcard (*) can be specified.

Auxiliary Object Class:
This field is only displayed if the **Structural Object Class** field is displayed. Select **YES** if auxiliary object classes can be selected in Mask 6c. Select **NO** if auxiliary object classes cannot be selected.

name The left part of the **name** field shows the naming attributes according to the schema and the selected structure rule. The administrator enter the DN of the object on the right-hand side of the mask.

If *operation* is **Display Objects**, a wildcard (*) can be specified for each name part. The only situations in which wildcards are permitted are as follows:

- At the beginning of a name part; for example, ***ith**
- At the end of a name part; for example, **Smi***
- At the beginning and end of a name part; for example, ***mi***

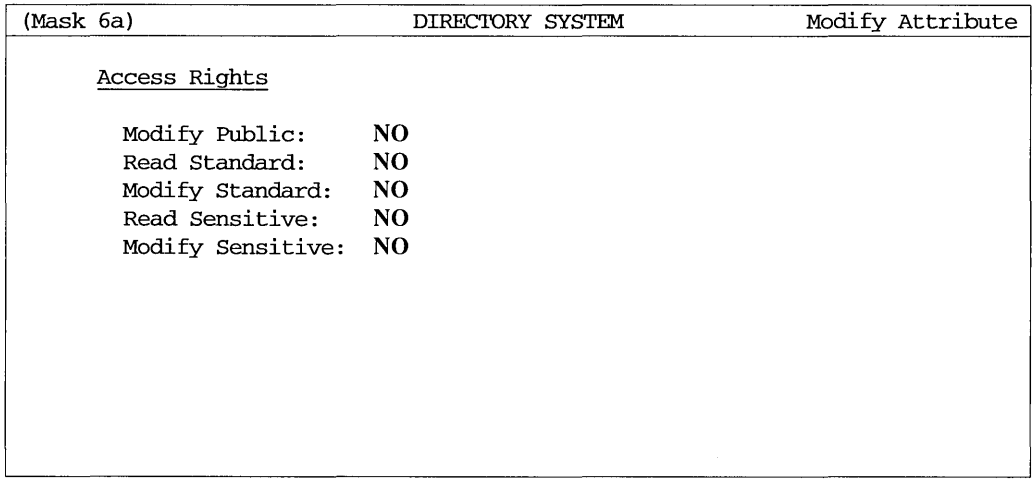
Wildcards that are displayed in the middle of name parts, (for example, **Smi*th**) are not considered to be wildcards. There is no special treatment of these characters if they appear in the middle of a name part.

In the case of all other operations, no wildcards can be entered and the object class of the object must be selected.

8.1.5 Mask 6a: Access Rights

Mask 6a (Figure 8-7) is displayed so that the **ACL** attribute can be modified. It is used to select the access rights to be entered in the ACL of the object selected in Mask 6 (Figure 8-6).

Figure 8–7. Mask 6a: Access Rights



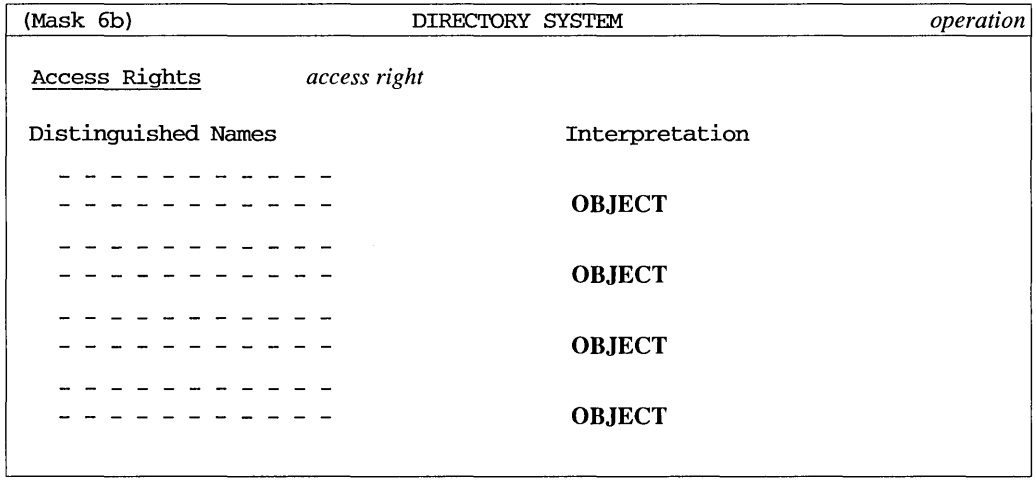
In Mask 6a, the *operation* field can only contain **Modify Attribute**.

To select the access rights to be entered in the ACL, toggle the appropriate **Access Rights** fields to **YES**.

8.1.6 Mask 6b: Authorization for Object Access

Mask 6b (Figure 8-8) is used to display and input up to four DNs for the access rights selected in Mask 6a.

Figure 8–8. Mask 6b: Authorization for Object Access



In Mask 6b, *operation* will be one of the following depending on the operation being performed:

Add Object
Display Objects
Modify Attribute

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The *access right* field will be one of the following depending on the access right selected in Mask 6a:

Modify Public
Read Standard
Modify Standard
Read Sensitive
Modify Sensitive

Mask 6b displays the following fields:

Distinguished Names
Enter a maximum of four DNs in the ACL for the access right selected. Separate the DNs with , (commas). Do not use a space before or after a comma.

For example:

/C=de/O=Smith Ltd/OU=dep1/CN=Huber,OU=AP11

The existence of an object with the specified DN will not be checked. This can cause problems if none of the DNs exist as objects. In this case, it is not possible to log into the DSA with one of the DNs specified. Therefore, nobody will have the access rights to change the object to which this ACL applies. In this case, the object for which no DN exists must be added to the DIT so that it is possible to log into the DSA using this DN, and to change the object to which the ACL applies.

Interpretation

Select one of the following values by pressing the space bar:

OBJECT To assign access to a single object

SUBTREE To assign access to a subtree (including subtree root)

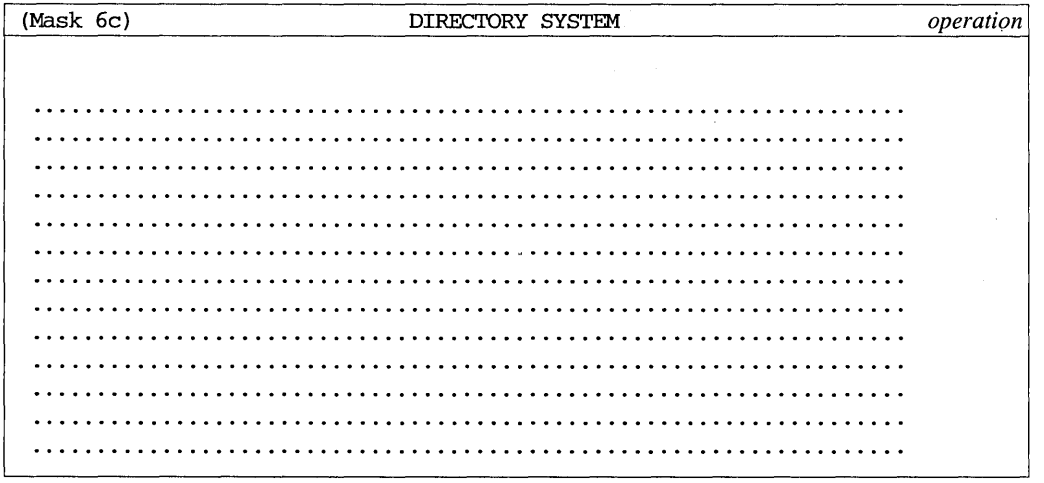
If the access rights are changed, the DNs according to the old ACL are displayed.

This value must be changed as required. If **SUBTREE** is selected, the specified access right is assigned to the entire subtree, including the subtree root.

8.1.7 Mask 6c: Auxiliary Object Class List

Mask 6c (Figure 8-9) is used to select one or more auxiliary object classes belonging to the selected structural object class.

Figure 8–9. Mask 6c: Auxiliary Object Class List



In Mask 6c, *operation* will be one of the following depending on the operation being performed:

Add Object
Display Objects
Add Attributes
Delete Attributes
Modify Attribute

Each line contains one auxiliary object class; for example, **MHS-User**. If the list contains more elements than can fit in a mask, use **<Scroll Up>** and **<Scroll Down>** to page up and down.

Use <↑> and <↓> to position the cursor on the element to be selected, and then press **<Return>** to mark the element. To unmark the positioned element, press **<Return>** again.

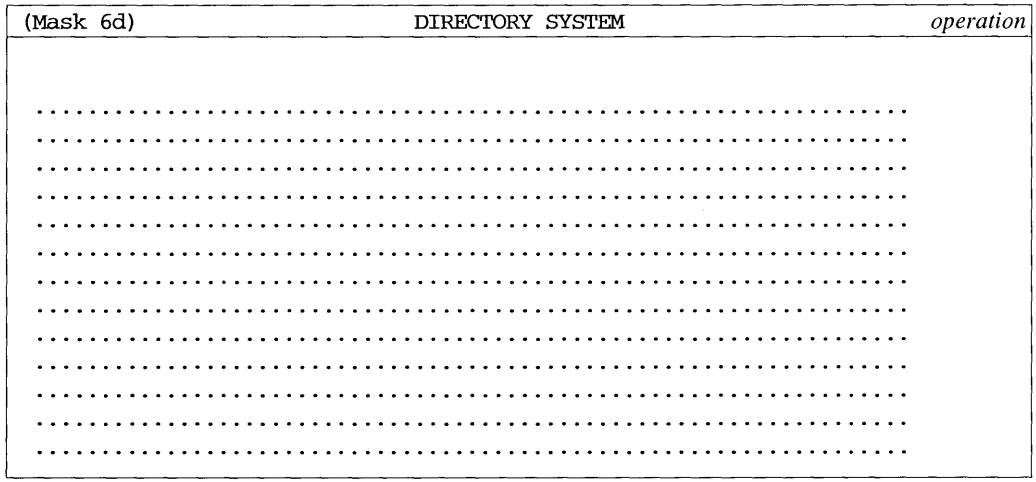
If you do not select any auxiliary object classes, it is assumed that all auxiliary object classes have been selected.

To exit from Mask 6c, press **<Del>**.

8.1.8 Mask 6d: Attribute List

Mask 6d (Figure 8-10) is used to select one or more attributes belonging to any structural and auxiliary object classes selected.

Figure 8–10. Mask 6d: Attribute List



In Mask 6d, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Delete Attributes
Modify Attribute
Modify Subtree

Each line contains one attribute; for example, **Access-Control-List**. In the case of the **Modify Attribute** and **Modify Subtree** operations, only one attribute can be selected from the attribute list.

In the case of the **Add Object** operation, mandatory attributes are automatically selected and are displayed at the beginning of the attribute list. Every mandatory recurring attribute reappears in the attribute list. This enables more than one attribute value to be entered for mandatory recurring attributes.

In the case of the **Add Attributes** operation, mandatory attributes are not displayed in the selection list because they are added to the object when it is created.

If the list contains more elements than can fit in a mask, use **<Scroll Up>** and **<Scroll Down>** to page up and down.

If the function called requires the selection of one or more elements, use <↑> and <↓> to position the cursor on the element to be selected, and then press **<Return>** to mark the element. If several recurring values for an attribute are to be handled (for example, in the **Add Object** and **Add Attributes** operations), press **<F8>** to mark that attribute. The attribute is then redisplayed in the mask if it is a valid recurring attribute and is selected. To unmark the positioned element, press **<Return>** again.

In the case of the **Add Object** operation, mandatory attributes are automatically selected; the cursor cannot be used to select or deselect these attributes.

8.1.9 Mask 7: Attributes

Mask 7 (Figure 8-11) is used to display and enter values for the attributes. Attribute names are displayed in the **Name** field.

Figure 8–11. Mask 7: Attributes

```
(Mask 7)                          DIRECTORY SYSTEM                          operation


  Attributes:

   Name    :  attribute name
   Value   :  - - - - - - - - - - - - - -
              - - - - - - - - - - - - - -

   Name    :  attribute name
   Value   :  - - - - - - - - - - - - - -
              - - - - - - - - - - - - - -

   Name    :  attribute name
   Value   :  - - - - - - - - - - - - - -
              - - - - - - - - - - - - - -
```

In Mask 7, *operation* will be one of the following depending on the operation being performed:

Add Object
Display Objects
Add Attributes

(See Section 8.2.3 for a description of the **Display Objects** operation.)

If *operation* is **Display Objects**, no input is accepted in this mask. In the case of **Add Object** and **Add Attributes** operations, the names of all the attributes selected in Mask 6d that do not require special masks are displayed in Mask 7. Only the values must be entered in this mask.

Mask 7 displays the following field:

Value Enter the value of the attribute. With character input, end with an ' (apostrophe); for example, **abc'**. (This makes it possible to use a space as the last character for attributes in which spaces at the end are allowed; for example, attributes with octet string syntax.)

With hexadecimal input, the character sequence must begin with an ' (apostrophe) and end with an ' (apostrophe); for example, **x'00FF'**.

The character input must conform to the attribute syntax on which the attribute type is based. See Appendixes C and E to determine the syntax on which an attribute type is based.

Leading and trailing spaces will be removed and spaces in the middle of an attribute value will be reduced to one space for the following syntaxes:

- **Numeric String**
- **Case Ignore String**
- **Case Exact String**
- **Printable String**
- **Case Ignore List**
- **Telephone Number Syntax**
- **Distinguished Name**

To add a default, local, or local/default DSA name in the DUA cache, **default'**, **local'**, or **default/local'** values must be entered respectively for the **DSA-Type** attribute.

If the syntax is **Boolean**, the value can be **TRUE** or **FALSE**.

If the syntax is **Preferred Delivery Method**, up to 10 integers (selected from a range of 0 to 9) can be entered. These integers are separated by a space.

If the syntax is **Distinguished Name**, the value name must be entered with no leading spaces.

For example:

/C=de/O=Smith Ltd/OU=dep1/CN=Huber,OU=AP11

The existence of an object with the DN entered is not checked.

The following attributes have their own masks:

- **CDS-Cell** (Mask 21)
- **Presentation-Address** (Mask 7a).
- **CDS-Replica** (Mask 22) (where CDS refers to **Cell Directory Service**)

Furthermore, the attributes with the following syntax also have their own masks:

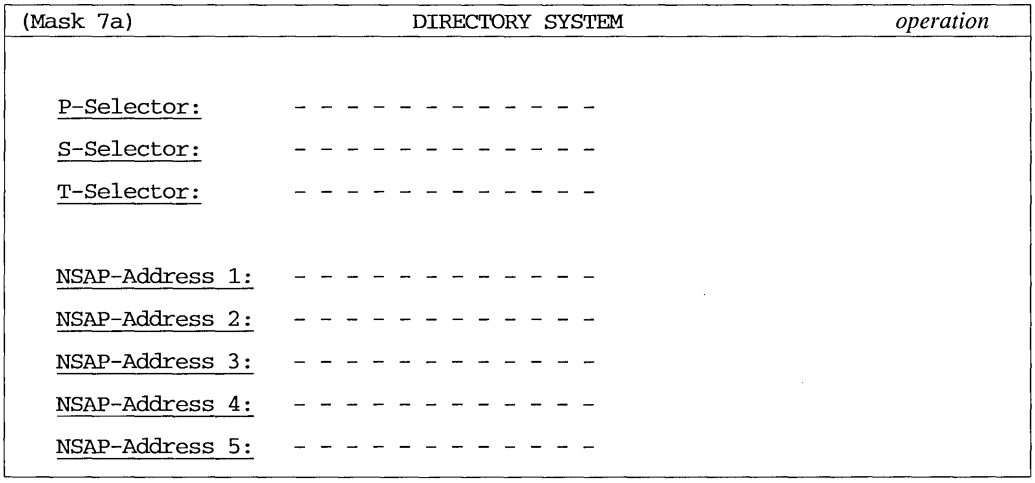
- TTX ID syntax (Mask 23)
- Telex Number syntax (Mask 24)
- Postal Address syntax (Mask 25)
- Fax Number syntax (Mask 26)
- MHS O/R Address syntax (Mask 27)
- MHS O/R Address syntax (Mnemonic) (Mask 28)
- MHS O/R Address syntax (Numeric) (Mask 29)
- MHS O/R Address syntax (Structured Postal) (Mask 30)
- MHS O/R Address syntax (Unstructured Postal) (Mask 31)
- MHS O/R Address syntax (Terminal) (Mask 32)
- MHS DL Submit Permission syntax (Mask 33)

- MHS DL Submit Permission syntax
  (Individual, Member of DL, Pattern Match) (Mask 34)
- MHS DL Submit Permission syntax (Member of Group) (Mask 35)
- MHS O/R Name syntax (Mask 34)

8.1.10 Mask 7a: Presentation-Address

Mask 7a (Figure 8-12) is used to display and enter the PSAP address.

Figure 8–12. Mask 7a: Presentation-Address



In Mask 7a, *operation* will be one of the following depending on the operation being performed:

Add Object
Display Objects
Add Attributes
Modify Attribute

(See Section 8.2.3 for a description of the **Display Objects** operation.)

A presentation address consists of a presentation selector (P-Selector) and a session address. A session address consists of a session selector (S-Selector) and a transport address. A transport address consists of a transport selector (T-Selector) and one or more network addresses.

Mask 7a displays the following fields:

P-Selector Enter the presentation selector.

S-Selector Enter the session selector.

T-Selector Enter the transport selector.

NSAP-Address
(Network Service Access Point Address)

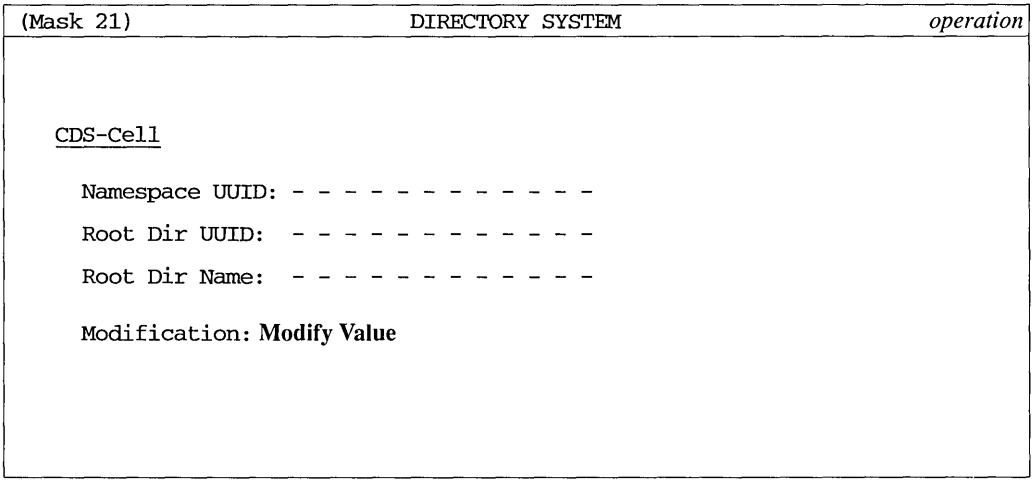
Enter the first NSAP address (**NSAP-Address 1**). This is required; other NSAP addresses are optional.

The syntax of the address fields is described in detail in Appendix D.

8.1.11 Mask 21: CDS-Cell

Mask 21 (Figure 8-13) is used to specify the **CDS-Cell** attribute.

Figure 8–13. Mask 21: CDS-Cell



In Mask 21, *operation* will be one of the following depending on the operation performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Note: Use the **cdscp show cell** command to generate this information. Then transfer it to Mask 21 by typing it in or by using a cut-and-paste operation. This means that an administrator does not require detailed knowledge of data formats or CDS concepts.

Mask 21 displays the following fields:

Namespace UUID
Enter the Universal Unique Identifier (UUID) of the CDS namespace. It is required to prevent ambiguities in CDS namespaces when a server manages more than one clearinghouse, and the clearinghouses are in different namespaces.

A CDS UUID consists of 16 hex digit pairs represented as 8 hexadecimal digits followed by a hyphen, 3 groups of 4 hexadecimal digits separated by hyphens, a hyphen, and 12 hexadecimal digits (for example, 01234567-89ab-cdef-0123-456789abcdef).

Root Dir UUID
Enter the UUID of the root directory. This parameter, in addition to the **Root Dir Name** parameter, is used to form the resolved and unresolved names in a CDS progress record.

Root Dir Name
Enter the root directory. This parameter, in addition to the **Root Dir UUID** parameter, is used to form the unresolved and resolved names in a CDS progress record. These parameters are only required when there are multiple cells contained in the CDS namespace.

Modification

Select one of the following values by pressing the space bar:

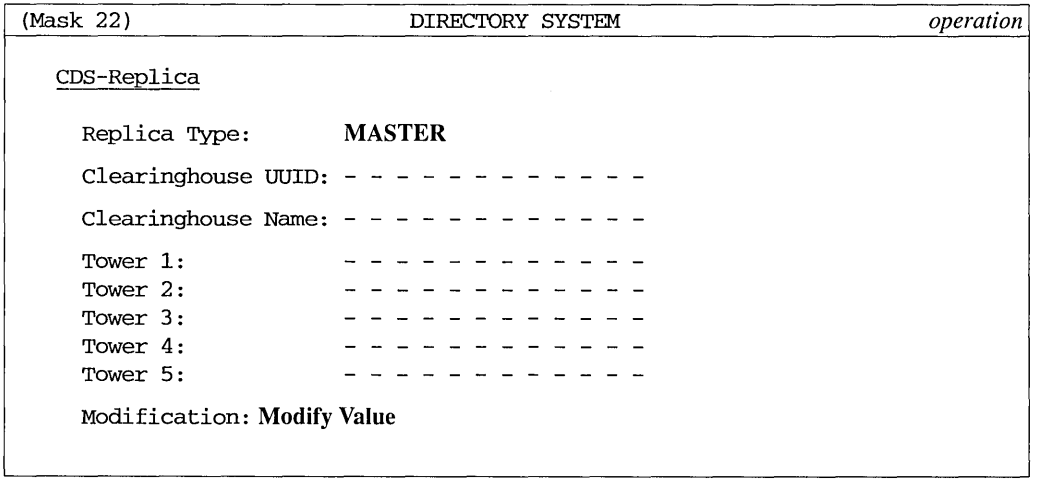
- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

CDS cell information is stored in two attributes: namely the **CDS-Cell** attribute and the **CDS-Replica** attribute. The information in one attribute complements the information in the other. Therefore, each object that contains a **CDS-Cell** attribute also needs to contain a **CDS-Replica** attribute.

8.1.12 Mask 22: CDS-Replica

Mask 22 (Figure 8-14) is used to specify the **CDS-Replica** attribute.

Figure 8–14. Mask 22: CDS-Replica



In Mask 22, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attribute
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Note: Use the **cdscp show cell** command to generate this information. Then transfer it to Mask 21 by typing it in or by using a cut-and-paste operation. This means that an administrator does not require detailed knowledge of data formats or CDS concepts.

Mask 22 displays the following fields:

Replica Type

Select **MASTER** for a modifiable replica or **READ ONLY** for a read-only replica. (**MASTER** can be changed to **READ ONLY**).

Clearinghouse UUID

Enter the clearinghouse UUID. The clearinghouse UUID never changes. Since the clearinghouse name can change, the clearinghouse UUID is used to verify the validity of the clearinghouse.

A CDS UUID consists of 16 hex digit pairs represented as 8 hexadecimal digits followed by a hyphen, 3 groups of 4 hexadecimal digits separated by hyphens, a hyphen, and 12 hexadecimal digits (for example, 01234567-89ab-cdef-0123-456789abcdef).

Clearinghouse Name

Enter the full name of the clearinghouse in which the replica is stored. This name is the name of a naming attribute that contains information on the last known address of the clearinghouse. This information enables the creation of an Remote Procedure Call (RPC) binding to the server that maintains the clearinghouse.

Tower Enter the tower set of the server that maintains the clearinghouse. A CDS tower contains addressing information and information on protocols supported by the clearinghouse server. The format of the tower value is the same format as a substring of the RPC string binding as follows:

```
protseq:netaddr
```

Modification

Select one of the following values by pressing the space bar:

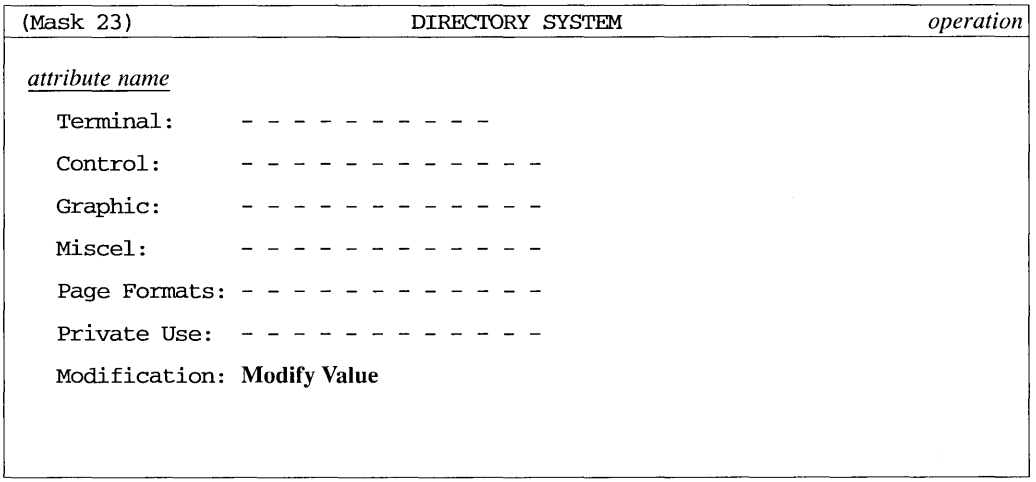
- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

CDS cell information is stored in two attributes: the **CDS-Cell** attribute, and the **CDS-Replica** attribute. The information in one attribute complements the information in the other. Therefore, each object that contains a **CDS-Cell** attribute also contains a **CDS-Replica** attribute.

8.1.13 Mask 23: Attribute with TTX-ID Syntax

Mask 23 (Figure 8-15) is used to specify the attribute name with TTX=ID syntax.

Figure 8–15. Mask 23: Attribute Name



In Mask 23, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Mask 23 displays the following fields:

attribute name
Displays the name of the attribute; for example, **TTX-Terminal-Identifier**.

Terminal Enter a unique terminal identifier for a teletex terminal. Enter the teletex nonbasic parameters in the remaining fields.

Control Enter the control character set for a teletex terminal (optional).

Graphic Enter the graphic character set for a teletex terminal (optional).

Miscel Enter the miscellaneous capabilities for a teletex terminal (optional).

Page Formats
Enter the page format for a teletex terminal (optional).

Private Use
Enter the **Private Use** parameters for a teletex terminal (optional).

Modification
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.14 Mask 24: Attribute with Telex Number Syntax

Mask 24 (Figure 8-16) is used to specify an attribute with Telex Number syntax.

Figure 8–16. Mask 24: Attribute with Telex Number Syntax

```
(Mask 24)                    DIRECTORY SYSTEM                    operation

attribute name

      Telex Number: - - - - - - - - - - - - - - -

      Country Code: - - - -

      Answerback:   - - - - - - - -

      Modification: Modify Value
```

In Mask 24, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Mask 24 displays the following fields:

attribute name
Displays the name of the attribute; for example, **Telex-Number**.

Telex Number
Enter the telex number of a telex terminal.

Country Code
Enter the country code of a telex terminal.

Answerback Enter the answerback code of a telex terminal.

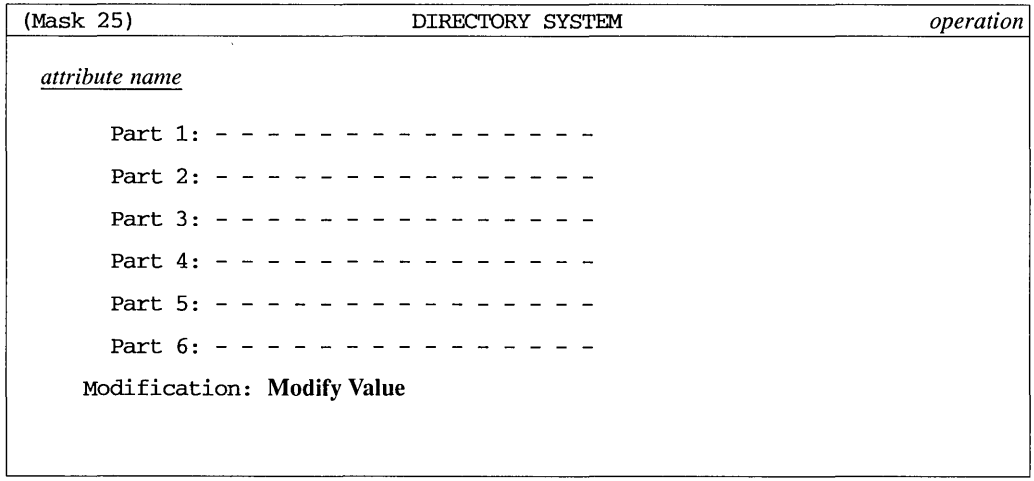
Modification
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.15 Mask 25: Attribute with Postal Address Syntax

Mask 25 (Figure 8-17) is used to specify an attribute with Postal Address syntax.

Figure 8–17. Mask 25: Attribute Name



In Mask 25, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Mask 25 displays the following fields:

attribute name
Displays the name of the attribute; for example, **Postal-Address**.

Part 1, Part2, ... Part 6
Enter at least one string. Normally, the postal address includes an addressee's name, street address, city, state or province, postal code, and possibly a post office box number.

Modification

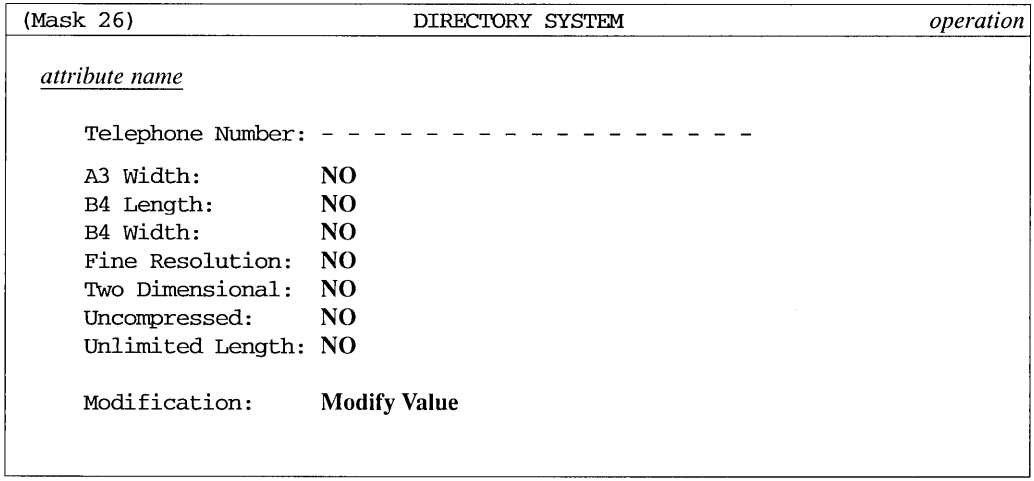
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.16 Mask 26: Attribute with Fax Number Syntax

Mask 26 (Figure 8-18) is used to specify an attribute name with Fax Number syntax.

Figure 8–18. Mask 26: Attribute Name



In Mask 26, the *operation* field can contain any of the following:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Mask 26 displays the following fields:

attribute name
: Displays the name of the attribute; for example, **Fax-Telephone-Number**.

Telephone Number
: Enter the telephone number for the facsimile telephone number.

fax parameters
: Select the G3 facsimile nonbasic parameters in the remaining fields. Select **YES** or **NO**. **NO** is the default value.

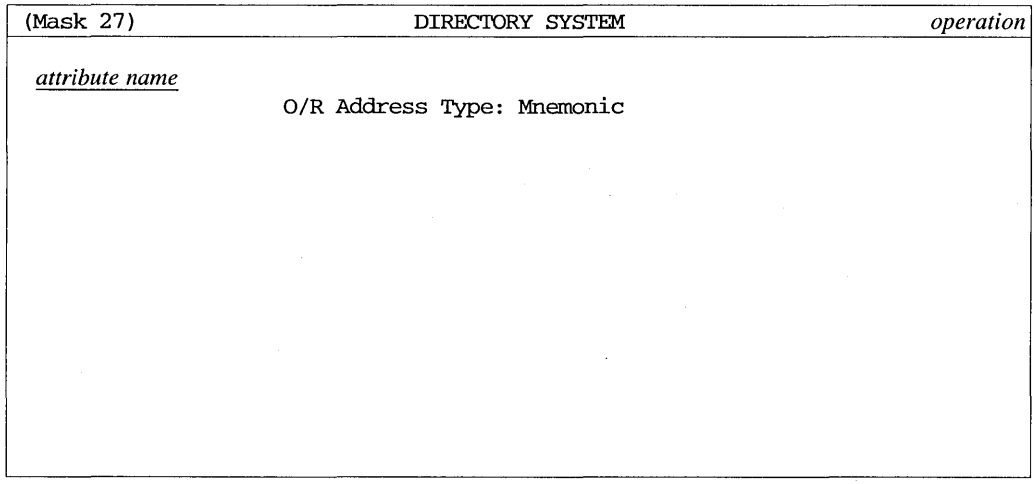
Modification
: Enter one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.17 Mask 27: Attribute with MHS O/R Address Syntax

Mask 27 (Figure 8-19) is used to specify an attribute with MHS O/R Address syntax.

Figure 8–19. Mask 27: Attribute with MHS O/R Address Syntax



Every user or distribution list (DL) is assigned one or more originator/recipient (O/R) addresses. An O/R address is an attribute list that distinguishes one user from another and identifies the user's point of access to the MHS or the DL's expansion point.

In Mask 27, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Modify Subtree

Mask 27 displays the following fields:

attribute name

Displays the name of the attribute; for example, **MHS-O/R-Address**.

O/R Address Type

Select one of the following values by pressing the space bar:

Mnemonic Displays Mask 28 for entering a mnemonic O/R address.

Numeric Displays Mask 29 for entering a numeric O/R address.

Structured Postal
Displays Mask 30 for entering a structured postal O/R address.

Unstructured Postal
Displays Mask 31 for entering an unstructured postal O/R address.

Terminal Displays Mask 32 for entering a terminal O/R address.

8.1.18 Mask 28: Attribute with MHS O/R Address Syntax (Mnemonic)

Mask 28 (Figure 8-20) is used to specify an attribute with MHS O/R Address syntax.

Figure 8–20. Mask 28: Attribute with MHS O/R Address Syntax (Mnemonic)

```
(Mask 28)                        DIRECTORY SYSTEM                        operation
 attribute name
                          O/R Address Type: Mnemonic

Country Name: - - -   ADMD Name: - - - - - - -  PRMD Name: - - - - - - - -
Org. Name: - - - - - - - - - - - - - - - - - - - - -
OU1: - - - - - - - - - - - - - - -  OU2: - - - - - - - - - - - - - - -
OU3: - - - - - - - - - - - - - - - - - - - - - - - - -
Com. Name: - - - - - - - - - - - - - - - - - - - - - -
Given Name:   - - - - - - - - - - - - - - - - -      Initials:  - - - - -
Surname:   - - - - - - - - - - - - - - - - - - - - -
Domain Type1: - - - - - - - -                Domain Type2:   - - - - - - - -
Domain Type3: - - - - - - - -                Domain Type4:   - - - - - - - -
Domain Value1: - - - - - - - - - - - - - - - - - - - -
Domain Value2: - - - - - - - - - - - - - - - - - - - -
Domain Value3: - - - - - - - - - - - - - - - - - - - -
Domain Value4: - - - - - - - - - - - - - - - - - - - -
       Modification: Modify Value
```

In Mask 28, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute.**

A mnemonic O/R address is one that mnemonically identifies a user or distribution list (DL). It identifies an administration management domain (ADMD) and a user or DL relative to it and is made up of the following attributes:

- One country name and one administration domain name, which together identify an ADMD
- One private domain name, one organization name, one organizational unit name, one personal name or common name, or a combination of the above; and, optionally, one or more domain-defined attributes, which together identify a user or DL relative to the ADMD.

Mask 28 displays the following fields:

attribute name
Displays the name of the attribute; for example, **MHS-O/R-Address.**

Country Name
Enter the name of the country of the ADMD that the **ADMD Name** field denotes.

ADMD Name
Enter an administration domain name that identifies an ADMD relative to the country denoted by the country name.

PRMD Name
Enter a private domain name that identifies a private management domain (PRMD). As a national matter, this identification may be either relative to the country denoted by a country name (so that PRMD names are unique within the country), or relative to the ADMD identified by an administration domain name.

Org. Name Enter an organization name that identifies an organization. As a national matter, this identification may be either relative to the country denoted by a country name (so that organization names are unique within the country), or relative to the management domain identified by a private domain name, or an administration domain name or both.

OU1, OU2, OU3, OU4
Enter the organizational unit names that identify one or more units (for example, divisions or departments) of the organization denoted in the *attribute name* field. Each unit, with the exception of the first, is a subunit of the units whose names precede it in the attribute.

Com. Name Enter a common name that identifies a user or distribution list.

Given Name
Enter the user's given name.

Initials Enter the initials of all of the user's names, with the exception of his or her surname.

Surname Enter the user's surname.

Generation Enter the user's generation; for example, **Jnr.**

Domain Type1, Domain Type2, ...
Enter the name of a class of information.

Domain Value1, Domain Value2, ...
Enter an instance of the class of information that the preceding **Domain Type** denotes.

Note: The widespread use of standard attributes produces more uniform and thus more user-friendly O/R addresses. However, it is anticipated that not all Management Domains (MDs) will be able to employ such attributes immediately. The purpose of domain-defined attributes is to permit an MD to retain its existing, native addressing conventions for a time. It is intended, however, that all MDs will migrate toward the use of standard attributes, and that domain-defined attributes will only be used for an interim period.

Modification

Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.19 Mask 29: Attribute with MHS O/R Address Syntax (Numeric)

Mask 29 (Figure 8-21) is used to specify an attribute with MHS O/R Address syntax.

Figure 8–21. Mask 29: Attribute with MHS O/R Address Syntax (Numeric)

```
(Mask 29)                        DIRECTORY SYSTEM                        operation

  attribute name
                        O/R Address Type: Numeric

Country Name: - - -    ADMD Name: - - - - - - - -    PRMD Name: - - - - -
Numeric User Identifier:  - - - - - - - - - - - - - - - - - - - - - - - -
Domain Type1: - - - - - - - -              Domain Type2:   - - - - - - - -
Domain Type3: - - - - - - - -              Domain Type4:   - - - - - - - -
Domain Value1: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Domain Value2: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Domain Value3: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Domain Value4: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


        Modification: Modify Value
```

In Mask 29, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

A numeric O/R address is one that numerically identifies a user. It identifies an administration management domain (ADMD), and a user relative to it.

A numeric O/R address is made up of the following attributes:

- One country name and one administration domain name, which together identify an ADMD
- One numeric user identifier and, conditionally, one private domain name, which together identify the user relative to the ADMD
- Conditionally, one or more domain-defined attributes that provide information in addition to that which identifies the user.

Mask 29 displays the following fields:

attribute name
: The name of the attribute is displayed in this field; for example, **MHS-O/R-Address**.

Country Name
: Enter the name of the country of the ADMD that the **ADMD Name** field denotes.

ADMD Name
: Enter an administration domain name that identifies an ADMD relative to the country denoted by the **Country Name** field.

PRMD Name
: Enter a private domain name that identifies a private management domain (PRMD). As a national matter, this identification may be either relative to the country denoted by a country name (so that PRMD names are unique within the country), or relative to the ADMD identified by an administration domain name.

Numeric User Identifier
: Enter a numeric user identifier that numerically identifies a user relative to the ADMD denoted by an administration domain name.

Domain Type1, Domain Type2, ...
Enter the name of a class of information.

Domain Value1, Domain Value2, ...
Enter an instance of the class of information that the **Domain Type** attribute denotes.

Note: The widespread use of standard attributes produces more uniform and thus more user-friendly O/R addresses. However, it is anticipated that not all Management Domains (MDs) will be able to employ such attributes immediately. The purpose of domain-defined attributes is to permit an MD to retain its existing, native addressing conventions for a time. It is intended, however, that all MDs will migrate toward the use of standard attributes, and that domain-defined attributes only will be used for an interim period.

Modification
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.20 Mask 30: Attribute with MHS O/R Address Syntax (Structured Postal)

Mask 30 (Figure 8-22) is used to specify an attribute with MHS O/R Address syntax.

Figure 8–22. Mask 30: Attribute with MHS O/R Address Syntax (Structured Postal)

```
(Mask 30)                    DIRECTORY SYSTEM                    operation
 attribute name
                    O/R Address Type: Structured Postal

Country Name: - - -    ADMD Name: - - - - - - - - -  PRMD Name: - - - - - - -
Postal Country Name: - - -          Postal Code: - - - - - - - - - - - - - -
Postal Address Details:          - - - - - - - - - - - - - - - - - - - - - -
Postal Delivery Point Name:      - - - - - - - - - - - - - - - - - - - - - -
Postal Delivery System Name:     - - - - - - - - - - - - - - - - - - - - - -
Postal General Delivery Address: - - - - - - - - - - - - - - - - - - - - - -
Postal Locale:                   - - - - - - - - - - - - - - - - - - - - - -
Postal Office Box Number:        - - - - - - - - - - - - - - - - - - - - - -
Postal Office Name:              - - - - - - - - - - - - - - - - - - - - - -
Postal Office Number:            - - - - - - - - - - - - - - - - - - - - - -
Postal Organization Name:        - - - - - - - - - - - - - - - - - - - - - -
Postal Patron Details:           - - - - - - - - - - - - - - - - - - - - - -
Postal Patron Name:              - - - - - - - - - - - - - - - - - - - - - -
Postal Street Address:           - - - - - - - - - - - - - - - - - - - - - -
      Modification: Modify Value
```

In Mask 30, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

A postal O/R address is one that identifies a user by means of a postal address.

Mask 30 displays the following fields:

attribute name
Displays the name of the attribute in this field; for example, **MHS-O/R-Address**.

Country Name

Enter the name of the country of the ADMD that the **ADMD Name** field denotes.

ADMD Name

Enter an administration domain name that identifies an ADMD relative to the country denoted by the **Country Name** field.

PRMD Name

Enter a private domain name that identifies a Private Management Domain (PRMD). As a national matter, this identification may be either relative to the country denoted by a country name (so that PRMD names are unique within the country), or relative to the ADMD identified by an administration domain name.

Postal Country Name

Enter the name of the country in which the user receives physical messages.

Postal Code

Enter the postal code for the geographical area in which the user receives physical messages. It identifies the area relative to the country denoted by the **Postal Country Name** attribute. Its values are defined by the postal administration of that country.

Postal Address Details

Enter additional information that may be useful for identifying the exact point at which the user receives physical messages; for example, room and floor numbers in a large building.

Postal Delivery Point Name

Identify the locus of distribution of the user's physical messages other than that denoted by the **Postal Office Name** attribute; for example, a geographical area.

Postal Delivery System Name

Enter the name of the postal delivery system through which the user is to receive physical messages.

Postal General Delivery Address
Enter the code that users give to the post office denoted by the **Postal Office Name** attribute for collection of the physical messages awaiting delivery to them.

Postal Locale
Identify the point of delivery of the user's physical messages other than that denoted by the **General Delivery Address**, **Postal Office Box Number**, or **Postal Street Address** attributes; for example, a building or a small village.

Postal Office Box Number
Enter the number of the post office box where the user receives physical messages. The box is located at the post office denoted by the **Postal Office Name** attribute.

Postal Office Name
The name of the municipality (for example, city or village) where the office in which the user receives physical messages is located.

Postal Office Number
Enter a number for the means of distinguishing between several post offices denoted by the **Postal Office Name** attribute.

Postal Organization Name
Enter the name of the postal organization through which the user receives physical messages.

Postal Patron Details
Enter any additional information required to identify the user for purposes of physical delivery; for example, the name of the organizational unit through which the user receives physical messages.

Postal Patron Name
Enter the name under which the user receives physical messages.

Postal Street Address
Enter the street address where the user receives physical messages; for example, 43 Primrose Lane.

Modification

Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.21 Mask 31: Attribute with MHS O/R Address Syntax (Unstructured Postal)

Mask 31 (Figure 8-23) is used to display an attribute with MHS O/R Address syntax.

Figure 8–23. Mask 31: Attribute with MHS O/R Address Syntax (Unstructured Postal)

```
(Mask 31)                      DIRECTORY SYSTEM                      operation
 attribute name
                      O/R Address Type: Unstructured Postal

Country Name: - - -  ADMD Name: - - - - - - - -  PRMD Name: - - - - - -
Postal Country Name: - - -          Postal Code: - - - - - - - - - -
Postal Address in Full: - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part1): - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part2): - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part3): - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part3): - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part4): - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part5): - - - - - - - - - - - - - - - - - - - -
Postal Address in Lines (Part6): - - - - - - - - - - - - - - - - - - - -
Postal Delivery System Name: - - - - - - - - - - - - - - - -
       Modification: Modify Value
```

In Mask 31, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects

Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Mask 31 displays the following fields:

attribute name
The name of the attribute is displayed in this field; for example, **MHS-O/R-Address.**

Country Name
Enter the name of the country of the ADMD that the **ADMD Name** field denotes.

ADMD Name
Enter an administration domain name that identifies an ADMD relative to the country denoted by the **Country Name** field.

PRMD Name
Enter a private domain name that identifies a private management domain PRMD. As a national matter, this identification may be either relative to the country denoted by a country name (so that PRMD names are unique within the country), or relative to the ADMD identified by an administration domain name.

Postal Country Name
Enter the name of the country in which the user receives physical messages.

Postal Code
Enter the postal code for the geographical area in which the user receives physical messages. It identifies the area relative to the country denoted by the **Postal Country Name** attribute. Its values are defined by the postal administration of that country.

Postal Address in Full
Enter the free-form and possibly multiline postal address of the user.

Postal Address in Lines (Part 1), (Part 2), ...
Enter the free-form postal address of the user in a sequence of printable strings, each representing a line of text.

Postal Delivery System Name
Enter the name of a postal delivery system through which the user receives physical messages.

Modification
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.22 Mask 32: Attribute with MHS O/R Address Syntax (Terminal)

Mask 32 (Figure 8-24) is used to display an attribute with MHS O/R Address syntax.

Figure 8–24. Mask 32: Attribute with MHS O/R Address Syntax

```
(Mask 32)                          DIRECTORY SYSTEM                          operation
 attribute name
                          O/R Address Type: Terminal

ISDN Number: - - - - - - - - - - - - - - -
ISDN Subaddress: - - - - - - - - - - - - - - - - -
Presentation Address: N
X121 Address: - - - - - - - - - - - - - - - -
Country Name: - - -   ADMD Name: - - - - - - - - -  PRMD Name: - - - - -
Terminal Identifier:  - - - - - - - - - - - - - - - - - -
Terminal Type: Not used
Domain Type1: - - - - - - - -                  Domain Type2:   - - - - - - - -
Domain Type3: - - - - - - - -                  Domain Type4:   - - - - - - - -
Domain Value1: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Domain Value2: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Domain Value3: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Domain Value4: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
       Modification: Modify Value
```

In Mask 32, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

A terminal O/R address identifies a user by means of the network address and, if required, the type of terminal. It may also identify the ADMD through which that terminal is accessed. In the case of a telematic terminal, it specifies the terminal's network address and, if applicable, its terminal identifier and terminal type. In the case of a telex terminal, it specifies its telex number.

A terminal O/R address is made up of the following attributes:

- One **Network-Address**
- Conditionally, one **Terminal-Identifier**
- Conditionally, one **Terminal-Type**
- Conditionally, both one **Country-Name** and one **Administration-Domain-Name** which together identify an ADMD.
- Conditionally, one **Private-Domain-Name** and, conditionally, one or more domain-defined attributes, all of which provide additional information for identification of the user.

The presence of the **Private-Domain-Name** and the domain-defined attributes depends on the presence of the **Country-Name** and the **Administration-Domain-Name** attributes.

Mask 32 displays the following fields:

attribute name
The name of the attribute is displayed in this field; for example, **MHS-O/R-Address**.

ISDN Number
Enter the ISDN number of the user's terminal.

ISDN Subaddress

Enter the ISDN subaddress of the user's terminal, if any.

Presentation Address

If you select, **YES**, Mask 7a is displayed for entering the presentation address of the user's terminal. If you select **NO**, a presentation address is not included in the O/R address.

X121 Address

Enter the network address of the user's terminal.

Country Name

Enter the name of the country of the ADMD that the **ADMD Name** field denotes.

ADMD Name

Enter an administration domain name that identifies an ADMD relative to the country denoted by the **Country Name** field.

PRMD Name

Enter a private domain name that identifies a private management domain (PRMD). As a national matter, this identification may be either relative to the country denoted by a country name (so that PRMD names are unique within the country), or relative to the ADMD identified by an administration domain name.

Terminal Identifier

Enter the terminal identifier of a terminal; for example, a telex reply.

Terminal Type

Enter the terminal type.

Select one of the following options by pressing the space bar:

- **Not used** (default)
- **TELEX**
- **TELETEX**
- **G3 FACSIMILE**
- **G4 FACSIMILE**

- **IA5 TERMINAL**
- **VIDEOTEX**

Domain Type1, Domain Type2, ...
Enter the name of a class of information.

Domain Value1, Domain Value2, ...
Enter an instance of the class of information that the **Domain Type** field denotes.

Note: The widespread use of standard attributes produces more uniform and thus more user-friendly O/R addresses. However, it is anticipated that not all Management Domains (MDs) will be able to employ such attributes immediately. The purpose of domain-defined attributes is to permit an MD to retain its existing, native addressing conventions for a time. It is intended, however, that all MDs will migrate toward the use of standard attributes, and that domain-defined attributes will be used only for an interim period.

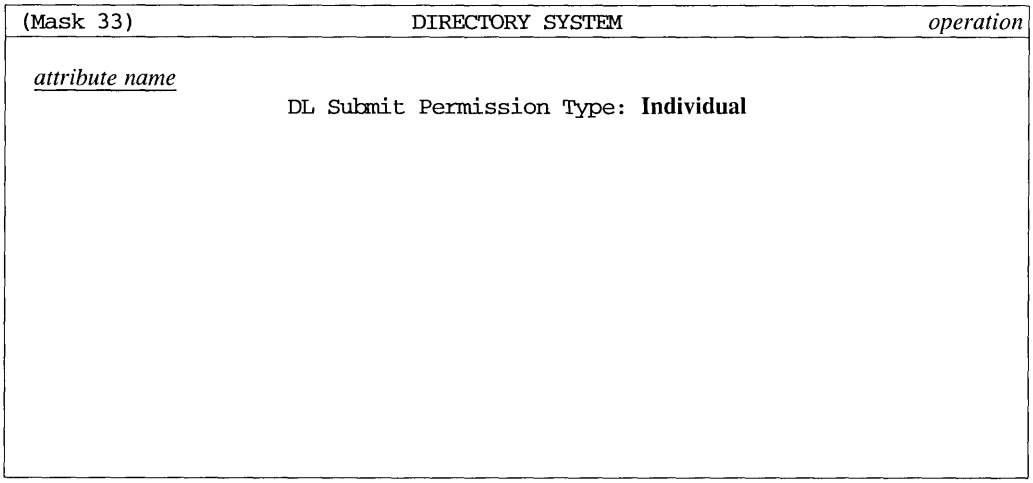
Modification
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.23 Mask 33: Attribute with MHS DL Submit Permission Syntax

Mask 33 (Figure 8-25) is used to specify an attribute with MHS DL Submit Permission syntax.

Figure 8–25. Mask 33: Attribute with MHS DL Submit Permission Syntax



In Mask 33, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Modify Subtree

Mask 33 displays the following fields:

attribute name
The name of the attribute is displayed in this field; for example, **MHS-DL-Submit-Permission**.

DL Submit Permission Type:
Select one of the following values by pressing the space bar:

Individual (default)
Displays Mask 34

Member of DL
Displays Mask 34

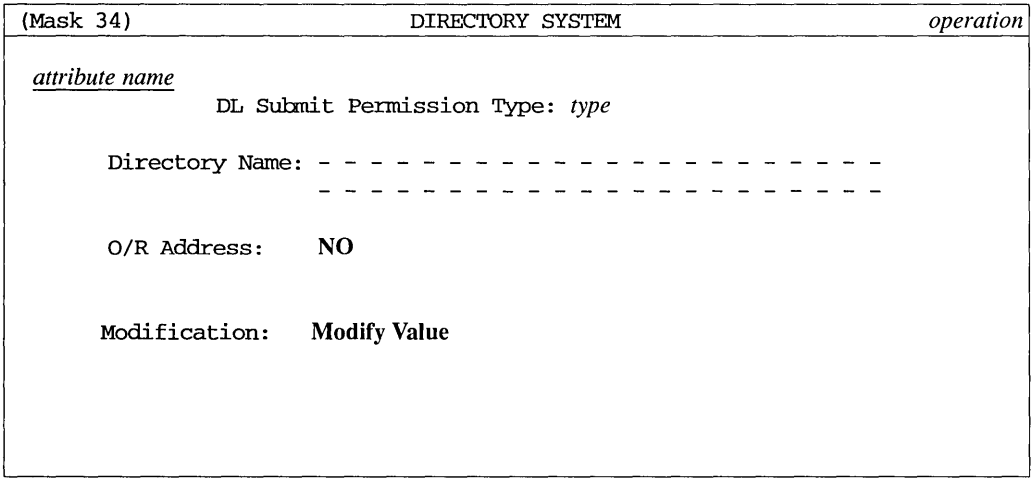
Pattern Match
Displays Mask 34

Member of Group
Displays Mask 35

8.1.24 Mask 34: Attribute with MHS O/R Name Syntax or MHS DL Submit Permission Syntax

Mask 34 (Figure 8-26) is used to specify an attribute with MHS O/R Name syntax.

Figure 8–26. Mask 34: Attribute with MHS O/R Name Syntax



In Mask 34, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute**.

Mask 34 displays the following fields:

attribute name

The name of the attribute is displayed in this field; for example, **MHS-DL-Members**.

DL Submit Permission Type
(if *operation* is **Modify Attribute**)

Select one of the following values by pressing the space bar:

- **Individual**
- **Member of DL**
- **Pattern Match**
- **Member of Group**

  When **Member of Group** is selected, Mask 35 is displayed.

Depending on the current value of the attribute, the default value can be one of the first three options. This field is a display field when the operation is **Add Object** or **Add Attribute**. It is a toggle field when the operation is **Modify Attribute**.

Directory Name

Enter the name assigned to the user or DL by the worldwide (X.500) directory. It has Distinguished Name syntax.

O/R Address

Select **YES** or **NO**. If **YES** is selected, O/R address masks are displayed to handle O/R addresses. **NO** is the default value. One of the two fields (**Directory Name**, **O/R Address**) must be present.

Modification

Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)

- **Add Value**
- **Delete Value**

8.1.25 Mask 35: Attribute with MHS DL Submit Permission Syntax

Mask 35 (Figure 8-27) is used to display an attribute with MHS O/R Name syntax.

Figure 8–27. Mask 35: Attribute with MHS O/R Name Syntax

```
(Mask 35)                    DIRECTORY SYSTEM                    operation

attribute name
              DL Submit Permission Type: Member of Group

   Member of Group: - - - - - - - - - - - - - - - - - - -
                    - - - - - - - - - - - - - - - - - - -



   Modification:    Modify Value
```

In Mask 35, *operation* will be one of the following depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

(See Section 8.2.3 for a description of the **Display Objects** operation.)

The **Modification** field is only shown when *operation* is **Modify Attribute.**

Mask 35 displays the following fields:

attribute name
The name of the attribute is displayed in this field; for example, **MHS-DL-Submit-Permission.**

DL Submit Permission Type
(if *operation* is **Modify Attribute**)

Select one of the following values by pressing the space bar:

- **Individual**
- **Member of DL**
- **Pattern Match**
- **Member of Group**

Member of Group
Enter each member of the group of names whose name is specified, or of each nested group of names recursively. It has Distinguished Name syntax.

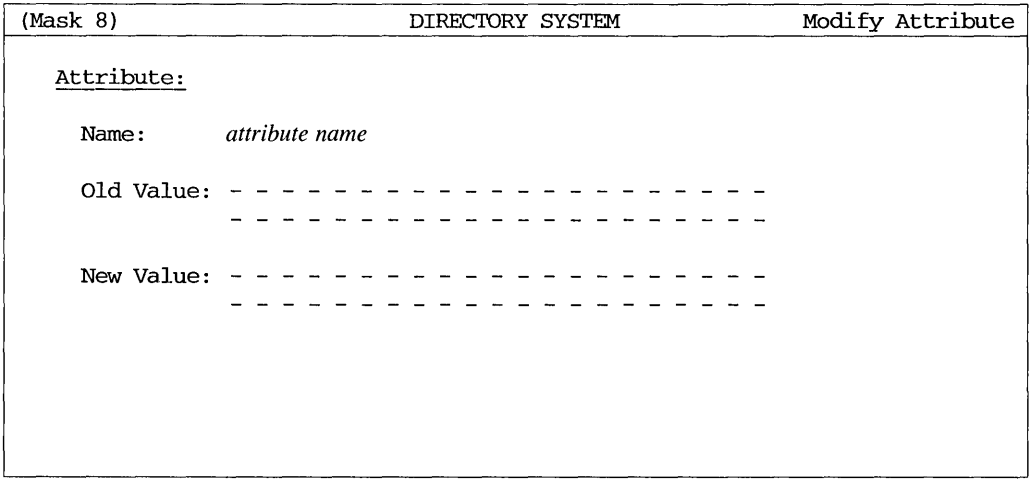
Modification
Select one of the following values by pressing the space bar:

- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.26 Mask 8: Attribute (Modify)

Mask 8 (Figure 8-28) is used to modify an attribute value that does not require a special mask.

Figure 8–28. Mask 8: Attribute (Modify)



In Mask 8, *operation* is **Modify Attribute**.

The name of the attribute selected in Mask 6d, which does not require a special mask, is displayed in the **Name** field.

Mask 8 displays the following fields:

Old Value, New Value

The attribute value is displayed in both fields. If the attribute has more than one value, use **<Scroll Down>** to page through the other values. Modify the displayed attributes as follows:

- To replace the old value with a new value, enter a new value in the **New Value** field.
- To delete the old value, leave the **New Value** field blank.
- To add a new value, leave the **Old Value** field blank.

If the syntax is Boolean, the value can be **TRUE** or **FALSE**.

If the syntax is Preferred Delivery Method, the integers are entered separated by a space.

If the syntax is Distinguished Name, the name must be entered with no leading spaces. Separate the DNs with commas. Do not use a space before or after a comma.

For example:

/C=de/O=Smith Ltd/OU=dep1/CN=Huber,OU=AP11

The existence of an object with the DN entered is not checked.

The following attributes have their own masks:

- **Presentation-Address** (Mask 7a)
- **CDS-Cell** (Mask 21)
- **CDS-Replica** (Mask 22)

The attributes with the following syntax also have their own masks:

- TTX ID syntax (Mask 23)
- Telex Number syntax (Mask 24)
- Postal Address syntax (Mask 25)
- Fax Number syntax (Mask 26)
- MHS O/R Address syntax (Mask 27)
- MHS O/R Address syntax (Mnemonic) (Mask 28)
- MHS O/R Address syntax (Numeric) (Mask 29)
- MHS O/R Address syntax (Structured Postal) (Mask 30)
- MHS O/R Address syntax (Unstructured Postal) (Mask 31)
- MHS O/R Address syntax (Terminal) (Mask 32)
- MHS O/R Name syntax (Mask 34)
- MHS DL Submit Permission syntax (Mask 34)
  (Individual, Member of DL, Pattern Match)
- MHS DL Submit Permission syntax (Member of Group) (Mask 35)

After one of these attributes is selected from the attribute list (Mask 6d), the mask containing the attribute value is displayed automatically.

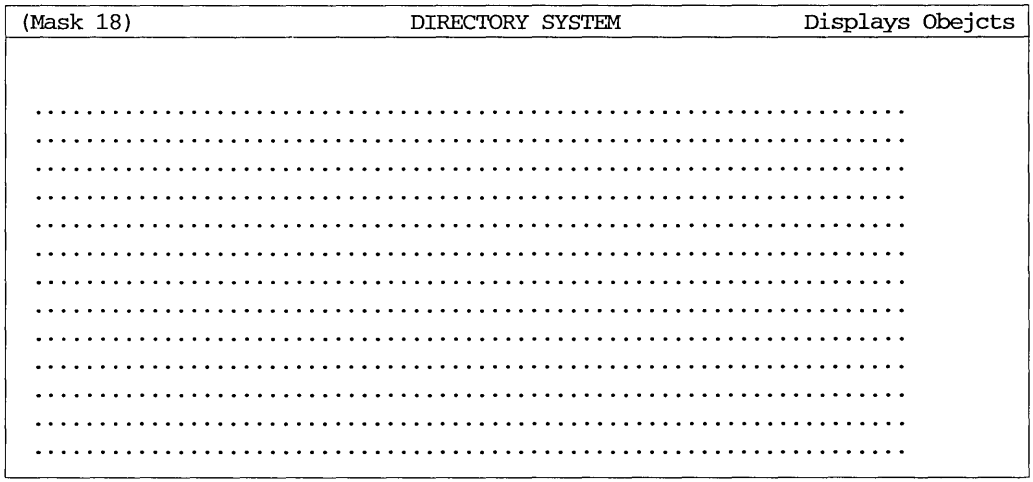
The last attribute value of a recurring attribute can only be deleted with the **Delete Attributes** function.

The first attribute value of a recurring attribute can only be added with the **Add Attributes** function.

8.1.27 Mask 18: Object List

Mask 18 (Figure 8-29) is used to display the results of display operations that cover more than one object.

Figure 8–29. Mask 18: Object List



In Mask 18, *operation* is **Display Objects**.

Each element is output in a line. If necessary, the line is shortened to the width of the mask; for example, **huber/ap11/Smith Ltd/de**.

If the list contains more elements than can fit in a mask, use **<Scroll Up>** and **<Scroll Down>** to page up and down.

If the function called requires the selection of one or more elements, use <↑> and <↓> to position the cursor on the element to be selected, and then press **<Return>** to mark the element. To unmark the positioned element, press **<Return>** again.

If none of the elements are selected, it is assumed that all of the elements have been selected.

To display a marked element in a detail mask (Masks 6, 6b, 7, 7a, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 34 or 35) press **<F1>**. Press **<F1>** again to return to the list display.

To exit from Mask 18, press **<Del>**.

8.2 Operations

Each of the following sections deals with one of the operations used for object administration.

8.2.1 Add Object

The Add Object operation adds a new object with attributes and access rights in the DIT. The object can be added to any node in the information tree.

Modify rights (ACLs) are required to the access class of the naming attribute in the parent object. The DN of the new object must not contain alias names in its name parts.

If no ACL attribute is specified, the ACL attribute of the parent node is added to the object. If no **Master Knowledge** attribute is specified, the current DSA is the master of the object.

If a shadow entry is to be added, a **Master Knowledge** attribute must be specified. Use shadow administration (see Chapter 10) to administer shadows. If the Add Object operation is used, a shadow entry can be added for which no master entry exists.

If a master entry is to be created under a shadow entry, the DUA also automatically adds a shadow entry under the master entry of the higher-level object. In this case, the DSA that is master of the higher-level object must be available.

Mask sequence

Mask 4 Select option number 1.

Mask 5 Select the structure rule of object to be added.

Mask 6 Enter the name and structural object class of the object to be added.

Select **YES** or **NO** in the **Auxiliary Object Class** field.

Mask 6c Select the auxiliary object classes. This mask is only displayed if **YES** is selected in the **Auxiliary Object Class** field of Mask 6.

Mask 6d Select the attributes for the object. Mandatory attributes are automatically displayed.

Enter the attribute values for the attributes that do not require a special mask in Mask 7.

If the attribute name or the syntax of the attribute selected is one of the following, it requires a special mask:

- **Access-Control-List** attribute (Mask 6b)
- **Presentation-Address** attribute (Mask 7a)
- **CDS-Replica** attribute (Mask 21)
- **CDS-Cell** attribute (Mask 22)
- **Telex Terminal** syntax (Mask 23)
- **Telex Number** syntax (Mask 24)
- **Postal Address** synatx (Mask 25)
- **Fax Number** syntax (Mask 26)
- **MHS O/R Address** syntax (Mask 27)

  Depending on what has been selected in the **O/R Address Type** field, one of the following masks is displayed:

  — Mnemonic O/R Address (Mask 28)

  — Numeric O/R Address (Mask 29)

  — Structured Postal O/R Address (Mask 30)

— Unstructured Postal O/R Address (Mask 31)

— Terminal O/R Address (Mask 32)

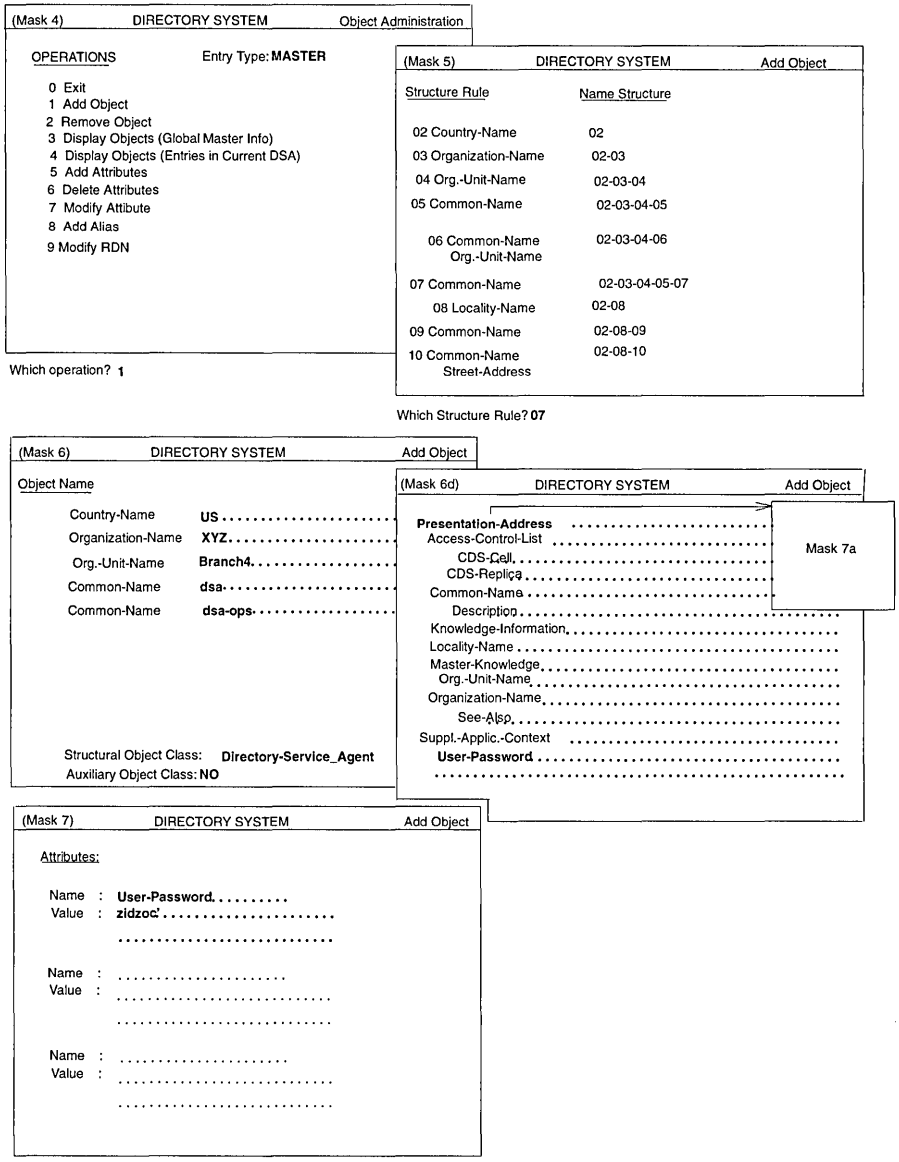
- **MHS DL Submit Permission** syntax (Mask 33)

  If **Individual**, **Member of DL**, or **Pattern Match** is selected in the **DL Submit Permission Type** field, then Mask 34 is displayed; otherwise, Mask 35 (Member of Group) is displayed.

- **MHS O/R Name** syntax (Mask 34)

Figure 8-30 shows the masks involved in a sample Add Objects operation. The administrator wants to add a DSA object to the DIT. User input is highlighted in bold type. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 8–30. Sample Add Object Operation



The administrator selects the Add Objects operation (option 1) from Mask 4 and presses **<Return>** or **<Menu>**.

Mask 5 is displayed and prompts the administrator for the structure rule number. Table A-2 in Appendix A shows the structure rule number for the structural object class DSA as 7. The administrator enters 7 at the Mask 5 prompt and presses **<Return>** or **<Menu>**. Mask 6 is displayed. The administrator toggles the values of the **Structural Object Class** field, using the space bar, to display **Directory-Service-Agent**. (Structure Rule 7 has five possible object classes: **Directory-Service-Agent**, **Application-Entity**, **MHS-Message-Store**, **MHS-Mess-Transfer-Agent**, and **MHS-User-Agent**.)

The left side of Mask 6 contains a list of the naming attributes for the object class **DSA**. The administrator enters the values for the naming attributes on the right side of Mask 6 and presses **<Return>** or **<Menu>**.

Mask 6d is displayed. Mask 6d contains a list of the mandatory and optional attributes of the object class **DSA** including all the attributes inherited from its superclasses. The mandatory attributes are highlighted, indicating that they have been selected automatically. To select optional attributes, scroll up and down using <↑> and <↓> to position the cursor on the element to be selected, and then press **<Return>** to mark the element. If the list contains more elements than can fit in a mask, use **<Scroll Up>** and **<Scroll Down>** to page up and down.

To define a DSA object in the DIT, the administrator needs to define the mandatory attributes of the **Directory-Service-Agent** class and those of its superclasses. The **Directory-Service-Agent** class has no mandatory attributes. The only mandatory attributes are **Common-Name** and **Presentation-Address**, which are inherited from the **Application-Entity** object class. It is not necessary to select **Common-Name**, because it has already been entered in Mask 6.

After the administrator presses **<Return>** or **<Menu>** to confirm the selections made from Mask 6d, Mask 7 displays the attributes that do not have their own masks. These attributes are presented three per screen. Figure 8-30 shows the optional attribute **User-Password** after the administrator has entered the attribute values. The administrator presses **<Return>** or **<Menu>** to confirm the input.

If the administrator had selected attributes in Mask 6d that have their own masks, such as **Presentation-Address** (Mask 7a), **CDS-Cell** (Mask 21) or **CDS-Replica** (Mask 22), these masks would be displayed in succession.

After the administrator completed the input for each attribute, the next mask would be displayed automatically. (Note that Figure 8-30 does not include any special masks).

8.2.2 Remove Object

The Remove Object operation removes an object from the DIT. Only objects on the end nodes of the DIT can be deleted.

Modify rights (ACLs) are required to the access class of the naming attribute in the object to be deleted.

The DN of the object must not contain alias names in its name parts. It is recommended that shadow administration be used to administer shadows (see Chapter 10).

If the master entry to be deleted is under a shadow entry, the DUA also automatically deletes the shadow entry under the master entry of the higher-level object. In this case, the DSA that is master of the higher-level object must be available.

Mask sequence

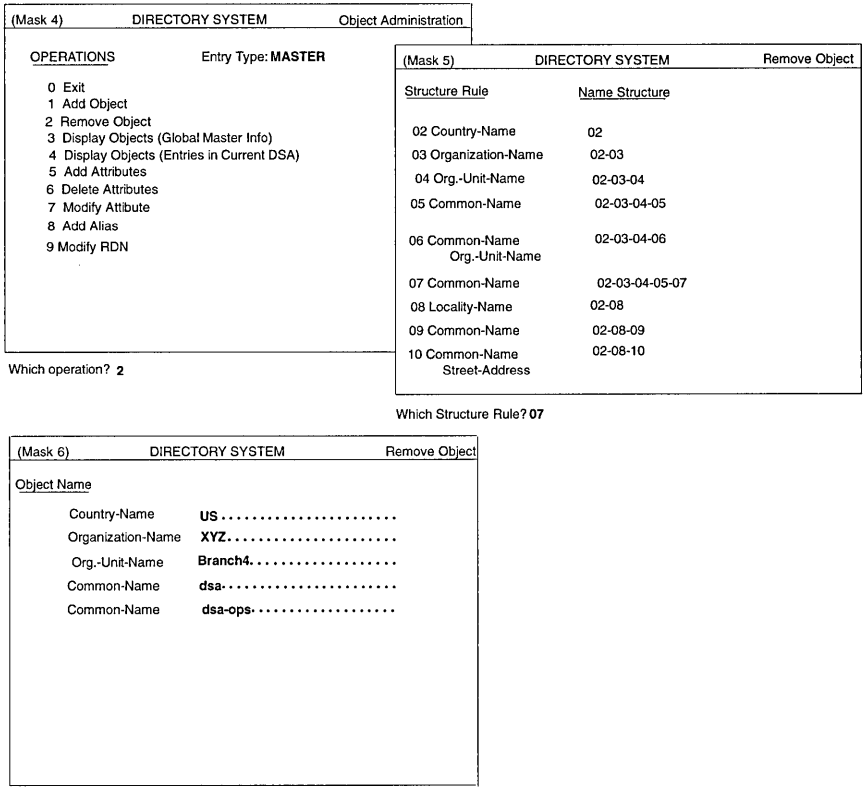
Mask 4 Select option number 2.

Mask 5 Enter the structure rule of object to be deleted.

Mask 6 Enter the DN of object to be deleted.

Figure 8-31 shows the masks involved in a sample Remove Objects operation. The administrator wants to remove a DSA object from the DIT. User input is highlighted in bold type. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 8–31. Sample Remove Object Operation



The administrator selects the Remove Objects operation (option number 2) from Mask 4 and presses **<Return>** or **<Menu>**.

Mask 5 is displayed and prompts the administrator for the structure rule number. Table A-2 in Appendix A shows the structure rule number for the structural object class **DSA** as 7. The administrator enters **7** at the Mask 5 prompt and presses **<Return>** or **<Menu>**. Mask 6 is displayed. The left side of Mask 6 contains a list of the naming attributes for the object class DSA. The administrator enters the values for the naming attributes on the right side of Mask 6 and presses **<Return>** or **<Menu>**.

8.2.3 Display Objects

The Display Objects operation displays either a single object and its attributes when the DN and the selected object class only match one object, or a list of objects and their attributes when the DN and the selected object class match more than one object.

If the alias objects also need to be displayed, an asterisk (*) must be specified for the object class.

The mask sequence in this section is used in the following three cases:

- When option 3 is selected in Mask 4 after logging into the DSA. This operation displays for the selected object or objects the master entries, which are stored in the various DSAs.
- When option 4 is selected in Mask 4 after logging into the DSA. This operation displays for the selected object or objects the entries (master and shadow) stored in the currently connected DSA.
- When option 3 is selected in Mask 4 after logging into the DUA cache. This operation displays for the selected objects all entries stored in the cache.

Mask sequence

Mask 4	Select option number 3.
Mask 5	Enter the structure rule of the objects to be displayed.
Mask 6	Enter the object name and object class of the objects to be displayed. Wildcards (*) can also be inserted in the different name parts and in the object class. Select **YES** or **NO** in the **Auxiliary Object Class** field.
Mask 6c	Select the auxiliary object classes. This mask is only displayed if **YES** is selected in the **Auxiliary Object Class** field of Mask 6. If more than one object is returned by the DSA, then Mask 18 is displayed.
Mask 18	Displays objects; select objects as required (see Section 8.1.27).
Mask 6	Displays the name of a selected object.
Mask 6b	Displays access rights to the selected object.
Mask 7	Displays an attribute of the selected object.
Mask 7a	Displays the presentation address of the selected object.
Mask 21	Displays the **CDS-Cell** attribute.
Mask 22	Displays the **CDS-Replica** attribute.
Mask 23	Displays the attribute with TTX ID syntax.
Mask 24	Displays the attribute with Telex Number syntax.
Mask 25	Displays the attribute with Postal Address syntax.
Mask 26	Displays the attribute with Fax Number syntax.
Mask 28	Displays the attribute with MHS O/R Address syntax (Mnemonic).
Mask 29	Displays the attribute with MHS O/R Address syntax (Numeric).
Mask 30	Displays the attribute with MHS O/R Address syntax (Structured Postal).

Mask 31 Displays the attribute with MHS O/R Address syntax (Unstructured Postal).

Mask 32 Displays the attribute with MHS O/R Address syntax (Terminal).

Mask 34 Displays the attribute with MHS DL Submit Permission syntax (Individual, Member of DL, Pattern Match).

Mask 34 Displays the attribute with MHS O/R Name syntax.

Mask 35 Displays the attribute with MHS DL Submit Permission syntax (Member of Group).

You can page up and down in Masks 6, 6b, 7, 7a, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 34 and 35 to display the objects read if more than one object is found.

The end of the list of selected objects is indicated by a message. During the Display Objects operation, you can move through the masks as follows:

In Masks 6, 6b, 7, 7a, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31 and 35:

<Return> or **<Menu>**
Displays the next attribute in Masks 6b, 7, 7a, 21, 22, 23, 24, 25, 26, 28 29, 30, 31, 32, 34 and 35, or, if the last attribute is displayed, displays the next object.

<Scroll Up>
Displays the name of the previous object in Mask 6.

<Scroll Down>
Displays the name of the next object in Mask 6.

<F1> Returns to the object list in Mask 18.

<Del> Aborts the operation and returns to Mask 4.

In Mask 32:

<Return> or **<Menu>**
Displays Mask 7a if **Presentation-Address** is present. Otherwise, the behavior is the same as in other masks.

<Scroll Up>
Displays the name of the previous object in Mask 6.

<Scroll Down>
Displays the name of the next object in Mask 6.

<F1> Returns to the object list in Mask 18.

<Del> Aborts the operation and returns to Mask 4.

In Mask 34:

<Return> or **<Menu>**
Displays Mask 28, 29, 30, 31 or 32 if **O/R Address** is present. Otherwise, the behavior is the same as in other masks.

<Scroll Up>
Displays the name of the previous object in Mask 6.

<Scroll Down>
Displays the name of the next object in Mask 6.

<F1> Returns to the object list in Mask 18.

<Del> Aborts the operation and returns to Mask 4.

Note: If the system cannot find an object, it displays one of the following messages:

```
ERROR: No objects found!
       To continue press <CR>
```

or

```
ERROR: Object (or superior object) doesn't exist!
       To continue press <CR>
```

<Scroll Up>, **<Scroll Down>**, and **<F1>** are selectable only if more than one object has been displayed.

8.2.4 Add Attributes

The Add Attributes operation is not available after the Logon to the DUA Cache operation is performed.

This operation adds one or more attributes to an object. At least one value must be entered for each attribute. The attributes to be added must be defined as attribute types in the AT.

Modify rights (ACLs) are required to the access class of the attribute to be added in the object.

The DN of the object must not contain alias names in its name parts. It is recommended that shadow administration be used to administer shadows (see Chapter 10).

If the master entry to be modified is under a shadow entry, the DUA also automatically changes the relevant shadow entry under the master entry of the higher-level object. If this DSA is not available, the master entry is modified, and the shadow entry must be modified by the administrator when the DSA is made available.

Mask sequence

Mask 4 Select option number 5.

Mask 5 Select the structure rule of the object.

Mask 6 Enter the name and structural object class of the object to be added.

Select **YES** or **NO** in the **Auxiliary Object Class** field.

Mask 6c Select the auxiliary object classes. This mask is only displayed if **YES** is selected in the **Auxiliary Object Class** field of Mask 6.

Mask 6d Select the attributes for the object. Mandatory attributes are automatically displayed.

Enter the attribute values for the attributes that do not require a special mask in Mask 7.

If the attribute name or the syntax of the attribute selected is one of the following, it requires a special mask:

- **Access-Control-List** attribute (Mask 6b)
- **Presentation-Address** attribute (Mask 7a)
- **CDS-Replica** attribute (Mask 21)
- **CDS-Cell** attribute (Mask 22)
- Telex Terminal syntax (Mask 23)
- Telex Number syntax (Mask 24)

- Postal Address synatx (Mask 25)
- Fax Number syntax (Mask 26)
- MHS O/R Address syntax (Mask 27)

  Depending on what has been selected in the **O/R Address Type** field, one of the following masks is displayed:

  — Mnemonic O/R Address (Mask 28)

  — Numeric O/R Address (Mask 29)

  — Structured Postal O/R Address (Mask 30)

  — Unstructured Postal O/R Address (Mask 31)

  — Terminal O/R Address (Mask 32)

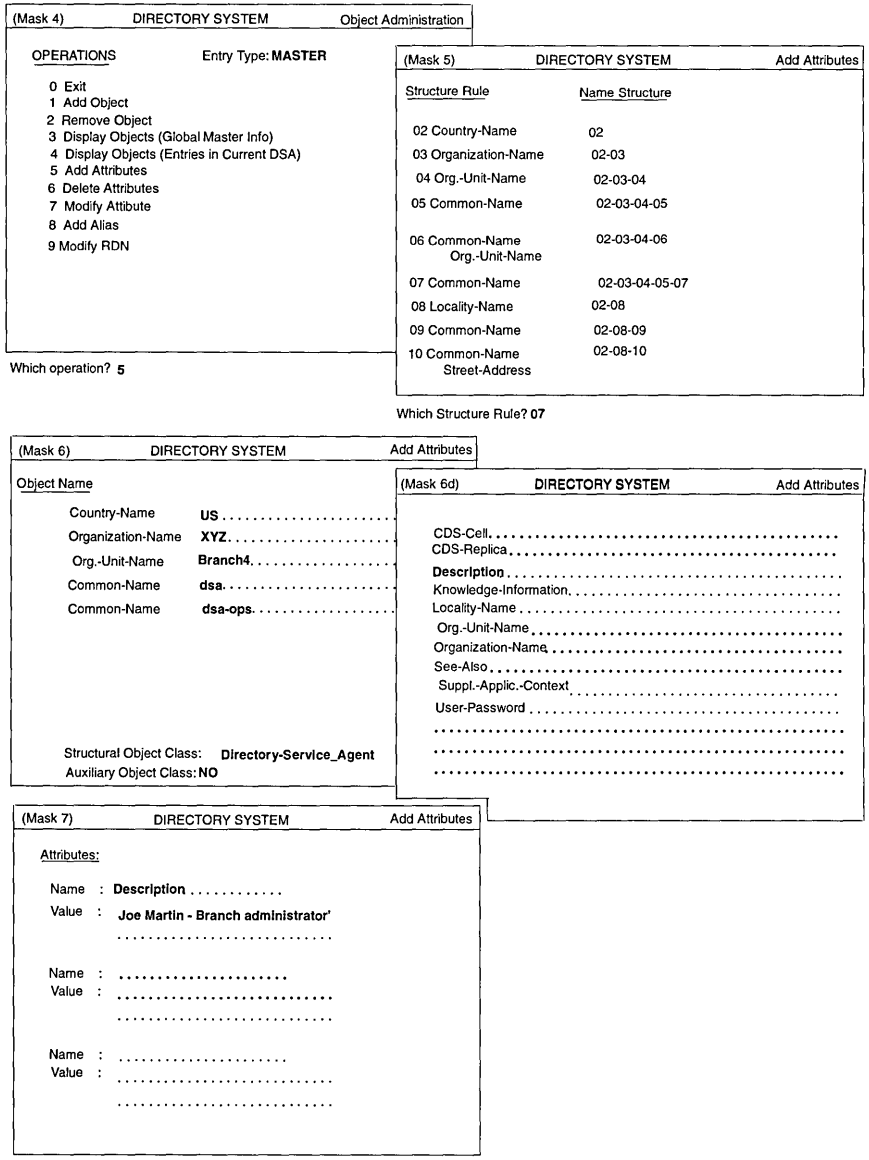
- MHS DL Submit Permission syntax (Mask 33)

  If **Individual**, **Member of DL**, or **Pattern Match** is selected in the **DL Submit Permission Type** field, then Mask 34 is displayed; otherwise, Mask 35 (Member of Group) is displayed.

- MHS O/R Name syntax (Mask 34)

Figure 8-32 shows the masks involved in a sample Add Attributes operation. The administrator wants to add the **Description** attribute of a DSA object to the DIT. Input is highlighted in bold type. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 8–32. Sample Add Attributes Operation



The administrator selects the Add Attributes operation (option number 5) from the Mask 4 menu and presses **<Return>** or **<Menu>**.

Mask 5 is displayed and prompts the administrator for the structure rule number. Table A-2 in Appendix A shows the structure rule number for the structural object class DSA as 7. The administrator enters **7** at the Mask 5 prompt and presses **<Return>** or **<Menu>**.

Mask 6 is displayed. The administrator toggles the values of the **Structural Object Class** field using the space bar to display **Directory-Service-Agent**. (Structure Rule 7 has five possible object classes: **Directory-Service-Agent**, **Application-Entity**, **MHS-Message-Store**, **MHS-Mess-Transfer-Agent**, and **MHS-User-Agent**).

The left side of Mask 6 contains a list of the naming attributes for the object class **DSA**. The administrator enters the values for the naming attributes on the right side of Mask 6 and presses **<Return>** or **<Menu>**.

Mask 6d is displayed. Mask 6d contains a list of the optional attributes of the object class **DSA** including all the attributes inherited from its superclasses. The administrator selects the new attribute, **Description**, by scrolling down using <↓> to position the cursor on the element to be selected, and then pressing **<Return>** to mark the element. (Several attributes may be selected. In this example, the administrator selects only one attribute. If attributes have their own masks, those masks are displayed after Mask 7.) The administrator presses **<Return>** or **<Menu>**.

Mask 7 is displayed with **Description**. The administrator enters the value for the new attribute and presses **<Return>** or **<Menu>** to confirm the input.

8.2.5 Delete Attributes

The Delete Attributes operation is not available after the Logon to the DUA Cache operation is performed.

This operation deletes one or more attributes of an object.

Modify rights (ACLs) are required to the access class of the attribute to be deleted in the object.

Mandatory attributes and naming attributes cannot be deleted. For example, the following attributes cannot be deleted:

- **Naming** attribute
- **Master Knowledge** attribute
- **ACL** attribute
- **Aliased Object** attribute

The DN of the object must not contain alias names in its name parts. It is recommended that shadow administration be used to administer shadows (see Chapter 10).

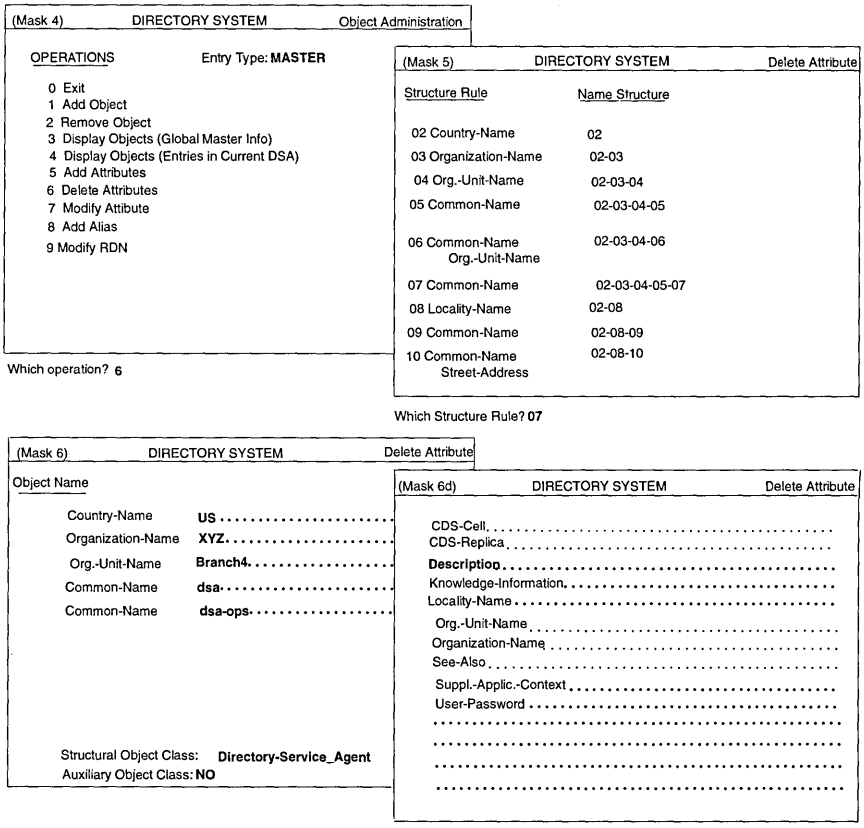
If the master entry to be modified is under a shadow entry, the DUA also automatically changes the relevant shadow entry under the master entry of the higher-level object. If this DSA is not available, the master entry is modified, and the shadow entry must be modified by the administrator when the DSA is made available.

Mask sequence

Mask 4	Select option number 6.
Mask 5	Select the structure rule of the object.
Mask 6	Enter the object name and object class. Select **YES** or **NO** in the **Auxiliary Object Class** field.
Mask 6c	Select the auxiliary object classes. This mask is only displayed if **YES** is selected in the **Auxiliary Object Class** field of Mask 6.
Mask 6d	Select the attributes to be deleted.

Figure 8-33 shows the masks involved in a sample Delete Attributes operation. The administrator wants to delete the **Description** attribute of a DSA object in the DIT. Input is highlighted in bold type. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 8–33. Sample Delete Attributes Operation



The administrator selects the Delete Attributes operation (option number 6) from Mask 4 and presses **<Return>** or **<Menu>**.

Mask 5 is displayed and prompts the administrator for the structure rule number. Table A-2 in Appendix A shows the structure rule number for the structural object class DSA as 7. The administrator enters **7** at the Mask 5 prompt and presses **<Return>** or **<Menu>**.

Mask 6 is displayed. The administrator toggles the values of the **Structural Object Class** field using the space bar to display **Directory-Service-Agent**. (Structure Rule 7 has five possible object classes: **Directory-Service-Agent**, **Application-Entity**, **MHS-Message-Store**, **MHS-Mess-Transfer-Agent**, and **MHS-User-Agent**).

The left side of Mask 6 contains a list of the naming attributes for the object class **DSA**. The administrator enters the values for the naming attributes on the right side of Mask 6 and presses **<Return>** or **<Menu>**.

Mask 6d is displayed. Mask 6d contains a list of the mandatory and optional attributes of the object class **DSA** including all the attributes inherited from its superclasses. The administrator selects the attribute to be deleted, **Description**, by scrolling down using <↓> to position the cursor on the element to be selected, and then pressing **<Return>** to mark the element. (Several attributes may be selected. In this example the administrator selects only one attribute.) The administrator presses **<Return>** or **<Menu>** to confirm the deletion.

8.2.6 Modify Attribute

The Modify Attribute operation is not available after the Logon to the DUA Cache operation is performed.

This operation modifies the value of an attribute in an object. Single value attributes can be modified, and recurring attribute values can be added, modified, and deleted.

Modify rights (ACLs) are required to the access class of the attribute to be modified in the object. If the object is an alias object, the ACL of the alias object parent is checked.

The DN of the object must not contain alias names in its name parts. It is recommended that shadow administration be used to administer shadows (see Chapter 10).

If the master entry to be modified is under a shadow entry, the DUA also automatically changes the relevant shadow entry under the master entry of the higher-level object. If this DSA is not available, the master entry is modified, and the shadow entry must be modified by the administrator when the DSA is made available.

Only one value of an attribute can be modified. To modify several attributes or attribute values, the operation has to be repeated.

The last attribute value of a recurring attribute can be deleted only with the Delete Attributes function.

Mask sequence

Mask 4 Select option number 7.

Mask 5 Select the structure rule of the object.

Mask 6 Enter the object name and object class.

Select **YES** or **NO** in the **Auxiliary Object Class** field.

Mask 6c Select the auxiliary object classes. This mask is only displayed if **YES** is selected in the **Auxiliary Object Class** field of Mask 6.

Mask 6d Select the attributes to be modified.

If the attribute name selected in Mask 6d is **Access Control List**, the **ACL** can be modified using Masks 6a and 6b (see the information that follows on Masks 6a and 6b). If the attribute name selected in Mask 6d is **Presentation-Address**, **CDS-Replica**, or **CDS-Cell**, or if the attribute syntax is TTX-ID, Telex Number, Postal Address, Fax Number, MHS O/R Address, MHS O/R Name or MHS DL Submit Permission then the first value is displayed in the following list of special masks. In order to modify the value, the value displayed is overwritten, and the type of modification is selected in the **Modification** field.

Mask 8 Enter the old or new value, or both, of the attribute to be modified in the case of attributes that do not require a special mask.

Mask 6a Select access rights that you want to modify.

Mask 6b The old ACL is displayed in this mask for all the different access rights. The values can then be overwritten. The DN fields as well as their interpretation fields can be modified, deleted, or assigned new entries.

Mask 7a Overwrite **Presentation-Address** displayed by new value.

Mask 21 Overwrite **CDS-Cell** displayed by new value.

Mask 22 Overwrite **CDS-Replica** displayed by new value.

Mask 23 Overwrite attribute value with **TTX ID** syntax displayed by new value.

Mask 24 Overwrite attribute value with **Telex Number** syntax displayed by new value.

Mask 25 Overwrite attribute value with **Postal Address** syntax displayed by new value.

Mask 26 Overwrite attribute value with **Fax Number** syntax displayed by new value.

Mask 28 Overwrite attribute value with **MHS O/R Address** syntax (**Mnemonic**) by new value.

Mask 29 Overwrite attribute value with **MHS O/R Address** syntax (**Numeric**) by new value.

Mask 30 Overwrite attribute value with **MHS O/R Address** syntax (**Structured Postal**) by new value.

Mask 31 Overwrite attribute value with **MHS O/R Address** syntax (**Unstructured Postal**) by new value.

Mask 32 Overwrite attribute value with **MHS O/R Address** syntax (**Terminal**) by new value.

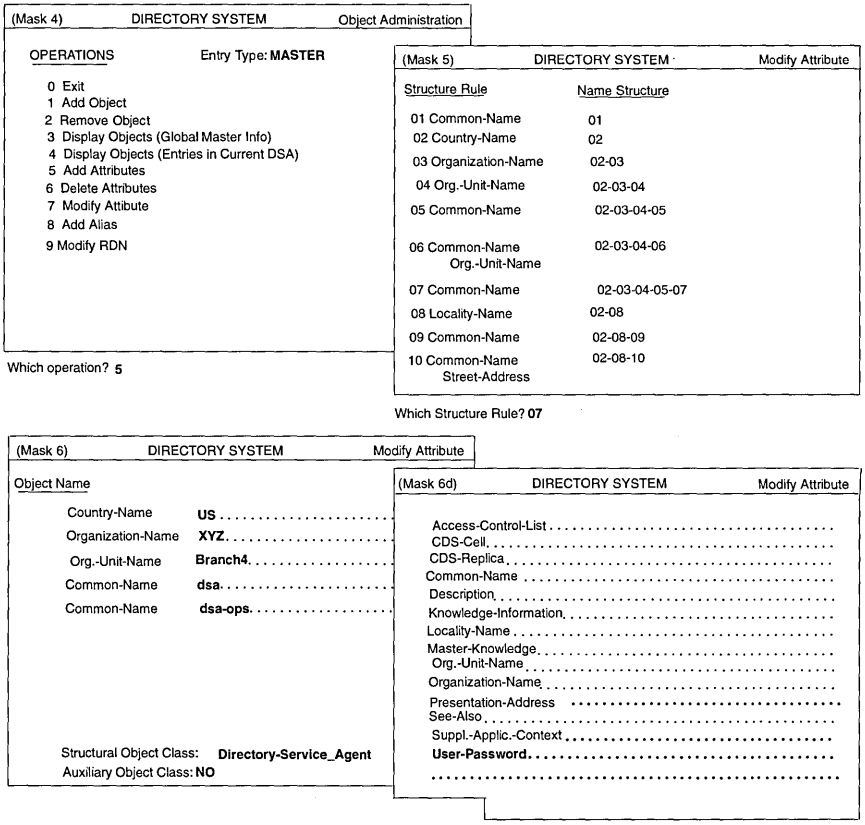
Mask 34 Overwrite attribute value with **MHS DL Submit Permission** syntax (**Individual**, **Member of DL**, **Pattern Match**) by new value.

Mask 34 Overwrite attribute value with **MHS O/R Name** syntax by new value.

Mask 35 Overwrite attribute value with **MHS DL Submit Permission** syntax (**Member of Group**) by new value.

Figure 8-34 shows the masks involved in a sample Modify Attribute operation.

Figure 8–34. Sample Modify Attribute Operation



The administrator wants to modify the **User-Password** attribute of a DSA object in the DIT. Administrator input is highlighted in bold type. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops.**

The administrator selects the Modify Attribute operation (option number 7) from Mask 4 and presses **<Return>** or **<Menu>**.

Mask 5 is displayed and prompts the user for the structure rule number. Table A-2 in Appendix A shows the structure rule number for the structural object class **DSA** as 7. The administrator enters **7** at the Mask 5 prompt and presses **<Return>** or **<Menu>**.

Mask 6 is displayed. The administrator toggles the values of the **Structural Object Class** field using the space bar to display **Directory-Service-Agent**. (Structure Rule 7 has five possible object classes: **Directory-Service-Agent, Application-Entity, MHS-Message-Store, MHS-Mess-Transfer-Agent,** and **MHS-User-Agent**).

The left side of Mask 6 contains a list of the naming attributes for the object class **DSA**. The administrator enters the values for the naming attributes on the right side of Mask 6 and presses **<Return>** or **<Menu>**.

Mask 6d is displayed. Mask 6d contains a list of the mandatory and optional attributes of the object class **DSA** including all the attributes inherited from its superclasses. The administrator selects the attribute to be modified, **User-Password**, by scrolling down using <↓> to position the cursor on the element to be selected, and then pressing **<Return>** to mark the element. Only one attribute can be modified.

After the attribute has been selected, the corresponding attribute is directly read from the DSA and either Mask 8 (for attributes without their own masks) or one of the specific attribute masks is displayed. The administrator then modifies the attribute value and presses **<Return>** or **<Menu>** to peform the Modify Attribute operation.

8.2.7 Add Alias

The Add Alias operation is not available after the Logon to the DUA Cache operation is performed.

This operation adds an alias for an object (aliased object) in the DIT. Alias objects are always end nodes in the DIT. The aliased object can be either an end node or an intermediate node in the DIT.

Modify rights (ACLs) are required to the access class of the naming attribute in the parent object of the alias object. When an attempt is made to modify the **Aliased-Object-Name** attribute using the Modify Attribute operation, the ACL of the parent object of the alias object is checked.

The DSA that is master of the parent object of the alias object is also master of the alias object.

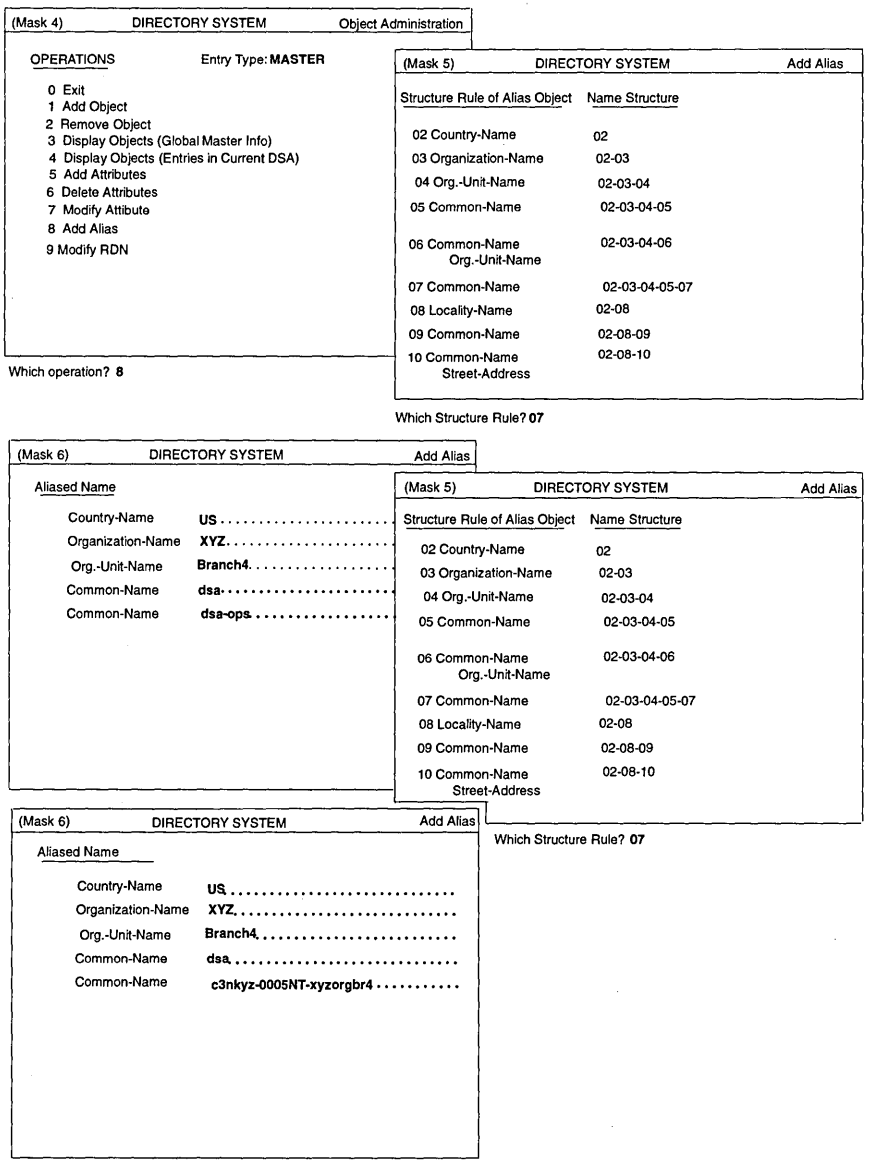
It is not checked whether or not the aliased object exists.

Mask sequence

Mask 4	Select option number 8.
Mask 5	Select the structure rule of the alias to be added.
Mask 6	Enter the name of the alias to be added.
Mask 5	Select the structure rule of the aliased object.
Mask 6	Enter the name of the aliased object.

Figure 8-35 shows the masks involved in a sample Add Alias operation. Input is highlighted in bold type. The administrator wants to add an alias of a DSA object to the Directory Information Tree. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=c3nkyz-0005NT-yxyorgbr4**. The administrator wants to create an alias with the common name **dsa-ops**.

Figure 8–35. Sample Add Alias Operation



The administrator selects the Add Alias operation (option number 8) from Mask 4 and presses **<Return>** or **<Menu>** .

Mask 5 is displayed and prompts the administrator for the structure rule number of the alias object. Table A-2 in Appendix A shows the structure rule number for the structural object class **DSA** as 7. The administrator enters **7** at the Mask 5 prompt and presses **<Return>** or **<Menu>**.

Mask 6 is displayed.

The administrator enters the alias object **dsa-ops** and confirms the input by pressing **<Return>** or **<Menu>**.

Mask 5 is displayed again and prompts for the structure rule number of the aliased object. The administrator enters **7**.

Mask 6 is displayed for entering the aliased object name. The administrator enters the new aliased name and confirms the input by pressing **<Return>** or **<Menu>**.

8.2.8 Modify RDN

The Modify RDN operation is not available after the Logon to the DUA Cache operation is performed.

This operation modifies the RDN of an object. Only objects on the end nodes of the DIT can be renamed.

Modify rights (ACLs) are required to the access class of the naming attribute in the object to be renamed.

The DN of the object must not contain alias names in its name parts. It is recommended that shadow administration be used to administer shadows (see Chapter 10).

If the master entry to be renamed is under a shadow entry, then the DUA also automatically renames the relevant shadow entry under the master entry of the higher-level object. In this case, the DSA that is master of the higher-level object must be available.

Mask sequence

Mask 4 Select option number 9.

Mask 5 Enter the structure rule of the object.

Mask 6 Enter the object name of object whose last RDN has to be changed.

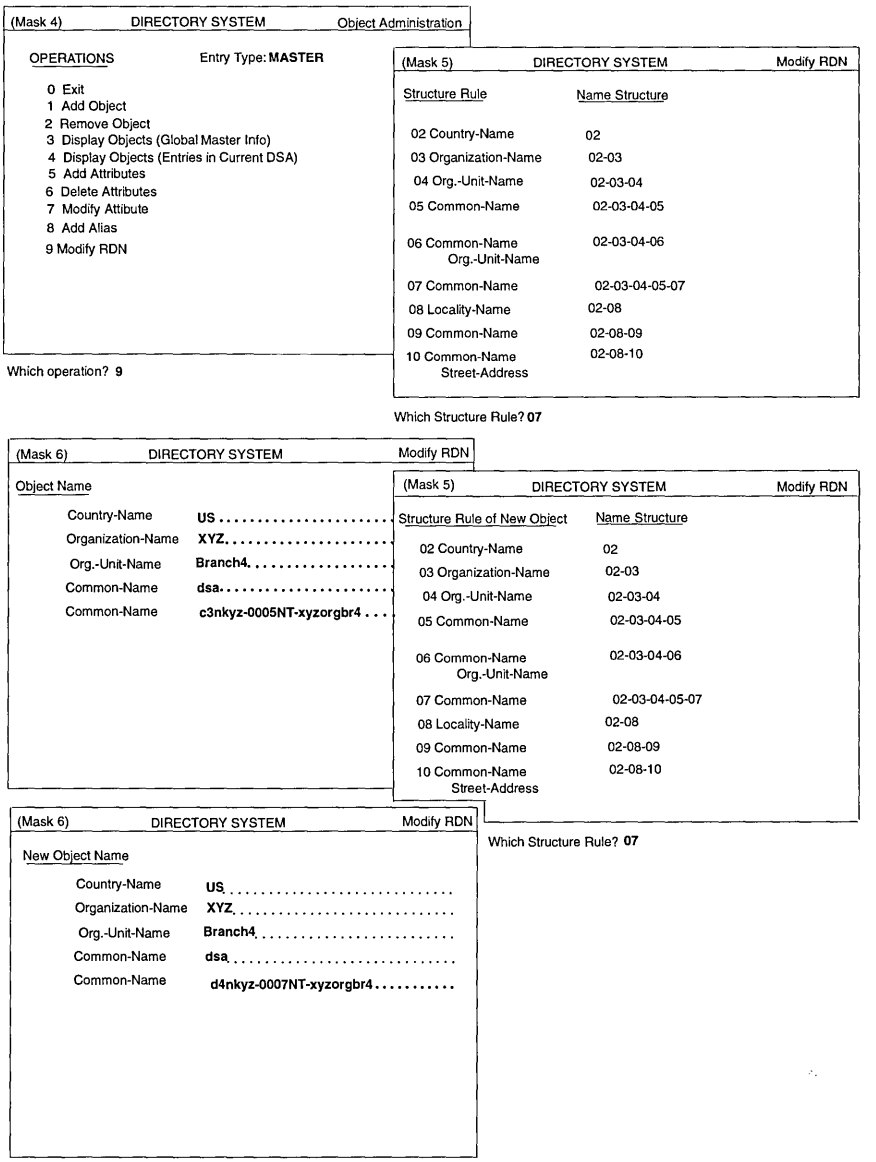
Mask 5 Enter the structure rule of the new object.

Note: Only structure rules that have the same parent rules as those of the old object can be selected.

Mask 6 Enter the new RDN.

Figure 8-36 shows the masks involved in a sample Modify RDN operation. Input is highlighted in bold type. The administrator wants to modify the RDN of a DSA object in the DIT. The DSA object has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=c3nkyz-0005NT-xyzorgbr4**. The administrator wants to modify the RDN **/CN=c3nkyz-0005NT-xyzorgbr4**.

Figure 8–36. Sample Modify RDN Operation



The administrator selects the **Modify RDN** operation (option number 9) from Mask 4 and presses **<Return>** or **<Menu>** .

Mask 5 is displayed and prompts the administrator for the structure rule number. Table A-2 in Appendix A shows the structure rule number for the structural object class DSA as 7. The administrator enters **7** at the Mask 5 prompt and presses **<Return>** or **<Menu>**.

Mask 6 is displayed. The administrator enters the values for the DSA object in Mask 6. The administrator presses **<Return>** or **<Menu>** to confirm the input.

Mask 5 is displayed again. The administrator enters the structure rule for the new object at the Mask 5 prompt.

Mask 6 is displayed. The system locks the first four fields so that they cannot be modified. The administrator changes the value of **Common-Name** to **d4nkyz-0007NT-xyzorgbr4**.

8.2.9 Display Local and Default DSA

It is only possible to select the Display Local and Default DSA operation after the Logon to the DUA Cache operation is performed and the Object Administration operation is selected.

This operation displays the DNs of the local and default DSAs as they are stored in the DUA cache.

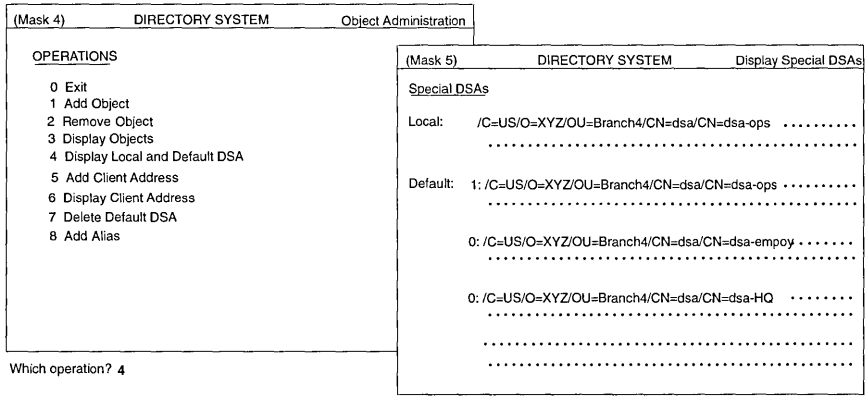
Mask sequence

Mask 4 Select option number 4.

Mask 4a Displays the DN of local and default DSAs.

Figure 8-37 shows how the DNs of the local and default DSAs stored in the DUA cache are displayed using Masks 4 and 4a.

Figure 8–37. Sample Display Local and Default DSA Operation



8.2.10 Add Client Address

It is only possible to select the Add Client Address operation after the Logon to the DUA Cache operation is performed and the Object Administration operation is selected.

This operation enters the client address of the C-Stub into the DUA cache.

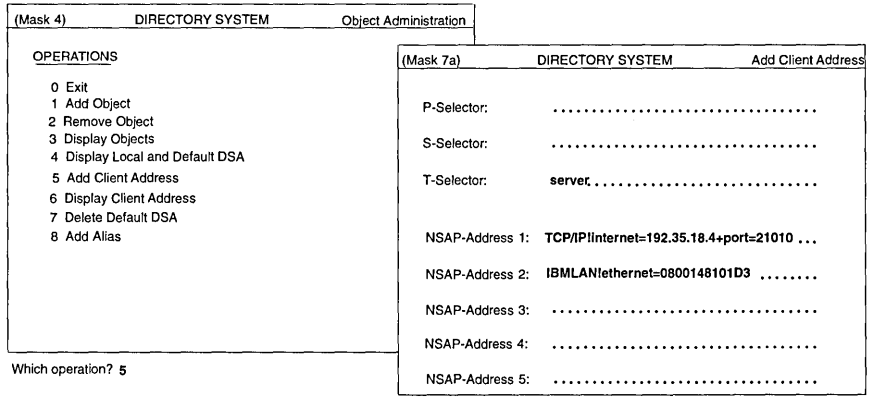
Mask sequence

Mask 4 Select option number 5.

Mask 7a Enter the presentation address.

Figure 8-38 shows how a client address is added to the DUA cache using Masks 4 and 7a. Input is highlighted in bold.

Figure 8–38. Sample Add Client Address Operation



8.2.11 Display Client Address

It is only possible to select the Display Client Address operation after the Logon to the DUA Cache operation is performed and the Object Administration operation is selected.

This operation displays the presentation address of the C-Stub.

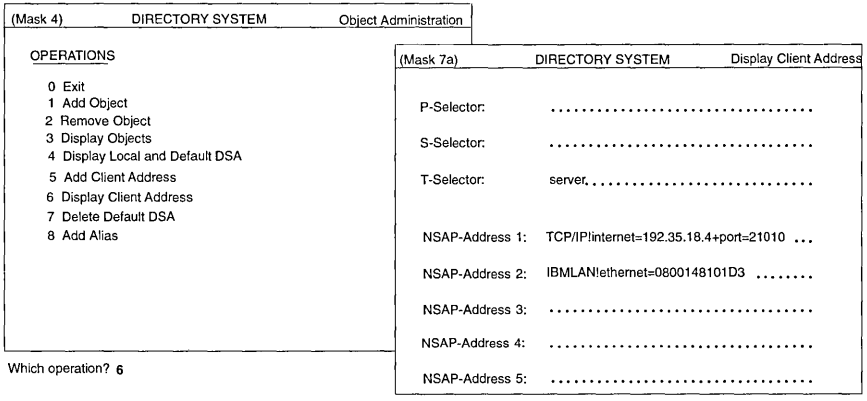
Mask sequence

Mask 4 Select option number 6.

Mask 7a Displays the presentation address.

Figure 8-39 shows how a client address stored in the DUA cache is displayed using Masks 4 and 7a.

Figure 8–39. Sample Display Client Address Operation



8.2.12 Delete Default DSA

It is only possible to select the Delete Default DSA operation after the Logon to the DUA Cache operation is performed and the Object Administration operation is selected.

This operation deletes a default DSA from the DUA cache.

Mask sequence

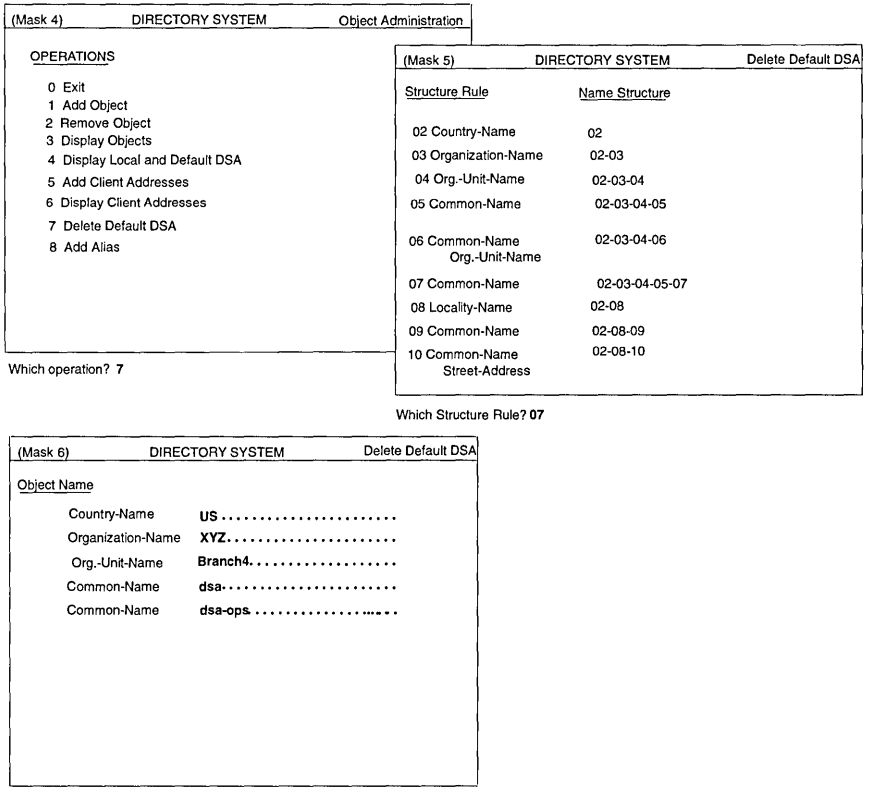
Mask 4 Select option number 7.

Mask 5 Select the structure rule of the default DSA.

Mask 6 Enter the DN of the default DSA to be deleted.

Figure 8-40 shows how to delete a default DSA that is stored in the DUA cache.

Figure 8–40. Sample Delete Default DSA Operation



Chapter 9

Schema Administration

Schema administration is a central administration task in a directory system and must be coordinated with all administrators in the system.

The directory schema governs the structure of the directory tree and consists of a set of rules that define the name structure and the object classes, as well as the attribute classes and their syntax. The X.500 directory standard describes only the schema concept and not the schema structure and protocol elements for the schema administration.

In GDS, the schema is stored as an object in the directory directly under the root with

DN : /CN=Schema

A default schema (see Appendixes A, B, and C) is supplied to initialize a directory service system.

After the GDS is configured and initialized, the DIT contains a schema object with a minimum number of ACLs. This means that anybody can change the schema object. Therefore, it is recommended that, after the object of the administrator is added, the administrator's DN be entered in the ACL of the schema object to prevent an unauthorized person from changing or destroying the schema object.

The recurring attributes described in the following list are the most important attributes of the directory schema object in GDS. These describe the database structure of the directory.

Structure Rule Table (SRT)

The SRT describes the structure of the DNs permitted in the directory and the object classes that can be assigned to objects with the name structure specified by the structure rule. Each structure rule determines the naming attributes for an object. The DN of the superior object is determined by the superior structure rule. The DN of the object is the DN of the superior object and an RDN whose attribute types are listed in the structure rule of the object. The following information is stored for each structure rule:

- Number of the structure rule
- Number of the superior rule
- Acronyms of the naming attributes
- Acronym of the assigned object class

The default entries in the SRT supplied with GDS are listed in Appendix A.

Object Class Table (OCT)

An object class defines the set of mandatory attributes that must be present in an entry of a given class, as well as the optional attributes that can be present in an entry of a given class.

The OCT describes the object classes supported by the directory (for example, **Country, Organizational-Person**). The following information is stored for each object class:

- Acronym of the object class
- Acronyms of superclasses
- Object identifier of the object class
- Name of the object class
- Kind of object class
- File number of the relevant object file

- Acronyms of auxiliary object classes
- Acronyms of mandatory attributes
- Acronyms of optional attributes

An object identifier can be defined by a registration authority such as the International Organization for Standardization (ISO) or by an individual for private purposes.

The default entries in the OCT supplied with GDS are listed in Appendix B.

Note: The objects can be stored in various object files depending on the file number component of the object class. There is usually one object file for the schema (file number 0) and three object files for all the other objects in the default schema (file numbers: 1, 2, and 3). If an object class is not stored in a file this is indicated by file number -1.

It is recommended that all object classes that have several attribute types in common be stored in the same object file in order to increase performance. Otherwise, they should be stored in different object files to save disk space.

Attribute Table (AT)

The AT describes the attribute types permitted in the directory. The following information is stored for each attribute type:

- Acronym of the attribute type
- Object identifier of the attribute type
- Attribute name
- Lower and upper bounds of the attribute value
- Maximum number of attribute values allowed
- Attribute syntax

- Flag indicating whether phonetic matching is permitted for the attribute
- Access class of the attributes Public, Standard, and Sensitive (see Section 1.5.3)
- Index level of the attribute

The default entries in the AT supplied with GDS are listed in Appendix C.

The following information is relevant for the attribute table:

- **Attribute Syntax**

  The attribute syntax defines the syntactic rules for the attribute values. It includes the data type in ASN.1 and, usually, one or more matching rules that are used for comparing values. (See Recommendation X.520 and X.411 for further information.)

  Table 9-1 shows the syntaxes that can be applied.

- **Phonetic Matching**

  If phonetic matching is permitted for an attribute, some slight deviations in the attribute value are also tolerated in search queries (for example, **Raier/Reier**) on request.

- **Index Level**

  The user can determine the priority of an attribute in search queries using the index level. GDS first searches for the entries whose attributes with a higher index level (that is, a higher number is specified) fulfill the **filter** condition. Attributes whose values seldom occur (for example, street and house number), which greatly restrict the number of hits, need to be given a high priority compared to attributes whose values frequently occur (for example, place of residence). Index level 0 must be assigned to the attributes that are never or seldom used as filter attributes.

  GDS first searches the directory for the objects according to the attributes with high priority in order to minimize the number of hits.

Table 9–1. Attribute Syntaxes

Syntax	Description
Case Exact String	A string is either a Printable String or a T.61 String (containing only T.61 characters according to X.208, T.61). Matches for equality and substrings. Matching is case sensitive. Uppercase/lowercase is observed.
Case Ignore String	A string is either a Printable String or a T.61 String (containing only T.61 characters according to X.208, T.61). Matches for equality and substrings. Matching is case sensitive. Uppercase/lowercase is not observed.
Octet String	Octet strings are any byte sequence (according to X.208). All characters are permitted. Matches for equality, substrings, and ordering.
Access Control List	Attribute values must correspond to the GDS specific attribute syntax. The ACL defines DNs and their interpretations (**SINGLE OBJECT** or **SUBTREE**) for different access classes.
Distinguished Name	Sequence of RDNs, which themselves are a set of AVAs; matches for equality.
Any syntax	Any byte sequence; no matching.
Object Identifier syntax	Attribute value must correspond to the Object identifier syntax (for example, 85.4.6 used as a hex value); matches for equality.
ASN.1 String	Attribute value is represented in ASN.1 format; no matching is defined.
Printable String	Only printable characters are permitted (according to X.208 standard); matches for equality, substrings, and ordering.

Syntax	Description
Numeric String	Only digits and space are permitted (according to X.208 standard); matches for equality and substrings (space characters are skipped during matching).
Case Ignore List	List of Case Ignore Strings.
Boolean syntax	Only TRUE and FALSE are permitted (according to X.208 standard); matches for equality.
Integer syntax	Only integers are permitted (according to X.208 standard); matches for equality and ordering.
UTC-Time	Absolute time (according to X.208 standard); matches for equality and ordering (for example, *YYMMDDhhmmssZ* (*YY* = year, *MM* = month, *DD* = day, *hh* = hour, *mm* = minutes, *ss* = seconds).
Telephone Number syntax	Telephone numbers in Printable String format; matches for equality and substrings (spaces, dashes, and slashes are skipped during comparison).
Search Guide syntax	Complex recursive attribute syntax; components are Object Identifier and Criteria; no matching.
Password syntax	Any byte sequence (according to X.208 standard); matches for equality only.
Country Name syntax	Same as Printable String syntax, but country names only according to ISO 3166 (for further information on Country syntax, see Appendix E); matches for equality.

Syntax	Description
Presentation Address	Consists of a Set of NSAP addresses and, optionally, P-Selector, S-Selector and T-Selector; matches for equality.
Fax Number syntax	Consists of a telephone number (Printable String) and, optionally, the G3 facsimile nonbasic parameters; matches for equality.
TTX-ID syntax	Consists of a teletex terminal ID and, optionally, the terminal nonbasic parameters; matches for equality.

Refer to Appendixes A, B, and C for examples of the respective tables and a description of the default schema.

The following examples illustrate which DNs can be stored in the DIT and which DNs do not conform to the definitions in the standard schema and, as a result, cannot be generated:

- Object 1, **/C=US/O=OSF/OU=Sales/CN=Smith**, with its object class equal to **ORP** conforms to the SRT and can be entered in the DIT. In contrast, object 2, **/C=US/OU=Sales**, with its object class equal to **OU** does not conform to the given structure rules.

  When adding object 1 to the DIT, it is necessary to specify at least the mandatory attributes as listed in the OCT under the entries for object class **ORP** and its superclasses **PER** and **TOP**. In this table, the attributes are **OCL**, **CN**, and **SN**. Furthermore, the optional attributes listed in the same table for the object class **ORP** and its superclasses **PER** and **TOP** can also be specified. The values of the attributes must conform to the lower and upper bound values, the maximum number of values, and the attribute syntax as specified in the AT.

- Object 3, **/C=US/O=OSF/OU=Sales/CN=Smith,OU=dep.3**, with four object class values, **ORP**, **PER**, **MUS**, and **TOP** conforms to all three tables and can be entered. In contrast, object 4, **/C=US/O=OSF/OU=Sales/CN=Smith,OU=dep.3**, with the object class value **APP** does not conform and is refused.

- Object 5, **/C=US/O=OSF/OU=Sales/CN=miller/CN=dsa-47**, with object class **APE** conforms to the tables only if the superior node,

/C=US/O=OSF/OU=Sales/CN=miller, has the object class value **APP**. It is refused, however, if the latter object class value is an **ORP**.

The operations of the schema administration are called by selecting option number 2 in Mask 3 (see Figure 7-3). The SRT, OCT, and AT are then loaded automatically in the memory of the administration program.

Modification of the schema is subject to several restrictions, which are described throughout this chapter.

It is advisable to save the old data before modifying the schema (see Section 4.2.1).

After the schema is modified, the initial DSA transfers the modified schema to all DSAs (if a shadowing job exists for the schema object). (See Chapter 10 for further information.)

9.1 Masks

This section describes the masks used for schema administration.

9.1.1 Mask 9: Schema Operations

Select the schema administration operation to be executed from Mask 9 (Figure 9-1).

Figure 9–1. Mask 9: Schema Operations

```
(Mask 9)                        DIRECTORY SYSTEM          Schema Administration

  Operations
        0   Exit
        1   Store Schema
        2   Load Schema
  (SRT):3   Display SRT
        4   Add SRT Entry
        5   Delete SRT Entry
        6   Modify SRT Entry
  (OCT):7   Display OCT
        8   Add OCT Entry
        9   Delete OCT Entry
       10   Modify OCT Entry
   (AT):11  Display AT
       12   Add AT Entry
       13   Delete AT Entry
       14   Modify AT Entry

Which operation ?
```

Mask 9 displays the following field:

Which operation
Select the number of the operation to be executed (0 to 14).

Depending on the selected operation, one of the following masks is displayed:

Store Schema
No mask

Load Schema
No mask

Display SRT
Mask 10 (Section 9.1.5)

Add SRT Entry
Mask 10 (Section 9.1.5)

Delete SRT Entry
Mask 10 (Section 9.1.5)

Modify SRT Entry
Mask 10 (Section 9.1.5)

Display OCT
Mask 11 (Section 9.1.6)

Add OCT Entry
Mask 11 (Section 9.1.6)

Delete OCT Entry
Mask 11 (Section 9.1.6)

Modify OCT Entry
Mask 11 (Section 9.1.6)

Display AT Mask 12 (Section 9.1.7)

Add AT Entry
Mask 12 (Section 9.1.7)

Delete AT Entry
Mask 12 (Section 9.1.7)

Modify AT Entry
Mask 12 (Section 9.1.7)

9.1.2 Mask 9a: Structure Rule List Mask

Mask 9a (Figure 9-2) is used to display more than one structure rule on the screen. For each structure rule, the rule number, superior rule number, and acronyms of naming attributes are displayed on one line.

When a structure rule is selected from this list, the structure rule and its remaining component (**Structural Object Class**) is displayed in Mask 10. For example:

9 4 CN OU

In the example **9** is the rule number, **4** is the superior rule number, and **CN** and **OU** are acronyms of naming attributes.

Figure 9–2. Mask 9a: Structure Rule List Mask

(Mask 9a)	DIRECTORY SYSTEM		Display SRT
1	0	CN	
2	0	C	
3	2	O	
4	3	OU	
5	4	CN	
6	4	CN	
7	4	CN	
8	4	CN	
9	4	CN	OU
10	7	CN	
11	7	CN	
12	7	CN	
13	7	CN	
14	7	CN	
15	2	L	
16	15	CN	

Each line contains one structure rule.

Use **<Scroll Up>** and **<Scroll Down>** to page up and down.

Use <↑> and <↓> to position the cursor on the element to be selected, and **<Return>** to mark the element. More than one element can be selected. To unmark the positioned element, press **<Return>** again.

Use **<Menu>** to display all the components of the selected structure rules.

If none of the structure rules is selected, it is assumed that all structure rules have been selected and they will be displayed one at a time in Mask 10.

To exit from Mask 9a, press **<Del>**.

9.1.3 Mask 9b: Object Class List Mask

Mask 9b (Figure 9-3) is used to display more than one object class entry on the screen. For each object class, the object class acronym, object class name, and object identifier are displayed on one line. For example:

ALI Alias 85.6.1

In the example, **ALI** is the object class acronym, **Alias** is the object class name, and **85.6.1** is the object identifier.

When an object class is selected from this list, additional components are displayed in Mask 11.

Figure 9–3. Mask 9b: Object Class List Mask

```
(Mask 9b)                    DIRECTORY SYSTEM                    Display OCT

ALI          Alias                          85.6.1
APE          Application-Entity             85.6.12
APP          Application-Process            85.6.11
C            Country                        85.6.2
DEV          Device                         85.6.14
DSA          Directory-Service-Agent        85.6.13
GON          Group-of-Names                 85.6.9
GTP          GDS-Top                        ---
LOC          Locality                       85.6.3
MDL          MHS-Distribution-List          86.5.1.0
MMS          MHS-Message-Store              86.5.1.1
MTA          MHS-Mess-Transfer-Agent        86.5.1.2
MUA          MHS-User-Agent                 86.5.1.4
MUS          MHS-User                       86.5.1.3
ORG          Organization                   85.6.4
ORP          Organizational-Person          85.6.7
```

Each line contains one object class.

Use **<Scroll Up>** and **<Scroll Down>** to page up and down.

Use <↑> and <↓> to position the cursor on the element to be selected, and **<Return>** to mark the element. More than one element can be selected.

To unmark the positioned element, press **<Return>** again.

Use **<Menu>** to display all the components of the selected object classes.

If none of the object classes is selected, it is assumed that all object classes have been selected and they will be displayed one at a time in Mask 11.

To exit from Mask 9b, press **<Del>**.

9.1.4 Mask 9c: Attribute List Mask

Mask 9c (Figure 9-4) is used to display more than one attribute entry on the screen. For each attribute, the attribute acronym, attribute name, and object identifier are displayed on one line. For example:

C Country-Name 85.4.6

In the example, **C** is the attribute acronym, **Country-Name** is the attribute name, and **85.4.6** is the object identifier.

When attributes are selected from this list, additional components are displayed in Mask 12.

Figure 9–4. Mask 9c: Attribute List Mask

(Mask 9c)	DIRECTORY SYSTEM	Display AT
ACL	Access-Control-List	43.12.2.1107.1.3.4.1
AON	Aliased-Object-Name	85.4.1
AT	Attribute-Table	43.12.2.1107.1.3.4.6
BC	Business-Category	85.4.15
C	Country-Name	85.4.6
CDC	CDS-Cell	43.12.2.1107.1.3.4.13
CDR	CDS-Replica	43.12.2.1107.1.3.4.14
CN	Common-Name	85.4.3
DI	Destination-Indicator	85.4.27
DSC	Description	86.4.13
FTN	Fax-Telephone-Number	85.4.23
IIN	Internat.-ISDN-Number	85.4.25
KNI	Knowledge-Information	85.4.2
L	Locality-Name	85.4.7
MDE	MHS-Deliverable-EITs	86.5.2.2
MDL	MHS-Deliv.-Cont.-Length	86.5.2.0

Each line contains a summary of one attribute.

Use **<Scroll Up>** and **<Scroll Down>** to page up and down.

Use <↑> and <↓> to position the cursor on the element to be selected, and **<Return>** to mark the element. More than one element can be selected.

To unmark the positioned element, press **<Return>** again.

Use **<Menu>** to display all the components of the selected attributes.

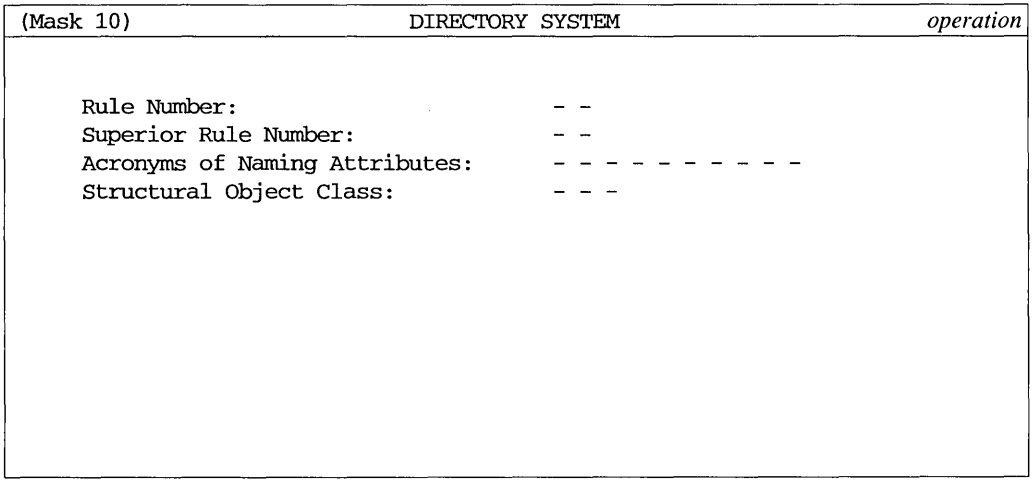
If none of the attributes is selected, it is assumed that all attributes have been selected and they will be displayed one at a time in Mask 12.

To exit from Mask 9c, press **<Del>**.

9.1.5 Mask 10: SRT Mask

Mask 10 (Figure 9-5) is used to display or enter the structure rules in the SRT.

Figure 9–5. Mask 10: SRT Mask



The *operation* field will be one of the following depending on the operation being performed:

Display SRT
Add SRT Entry
Delete SRT Entry
Modify SRT Entry

If *operation* is **Delete SRT Entry**, only **Rule Number** is displayed in the mask.

If *operation* is **Modify SRT Entry**, Mask 10 is displayed only with **Rule Number**. If the rule number that is entered is a correct one, Mask 10 is displayed again with all values belonging to that SRT entry. Afterward, all fields can be modified except **Rule Number** itself.

Mask 10 displays the following fields:

Rule Number Enter the number of the structure rule.

Superior Rule Number
Enter the number of the superior structure rule.

Acronyms of Naming Attributes
Enter up to three acronyms of naming attributes. These acronyms are separated by spaces. The naming attributes must exist in the AT.

Structural Object Class
Enter an object class that is assigned to this rule. It must exist as a structural object class in the OCT.

Function Keys

<Menu> Perform the selected operation (if the operation is **Display SRT**, display the next SRT element).

<Return> Go to the next field; if the field is the last one, perform the operation (if the operation is **Display SRT**, display the next SRT element).

<Scroll Down>
Display the next SRT element.

<Scroll Up>
Display the previous SRT element.

<Del> Cancel the operation; return to Mask 9.

9.1.6 Mask 11: OCT Mask

Mask 11 (Figure 9-6) is used to display or enter the Object Classes in the OCT.

Figure 9–6. Mask 11: OCT Mask

```
(Mask 11)                    DIRECTORY SYSTEM                    operation

     Object Class Acronym:        - - -
     Object Class Name:           - - - - - - - - - - - - - -
     Acronyms of Superclasses:    - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
     Object Identifier:           - - - - - - - - - - - - - -
     Object Class Kind:           Abstract
     File Number:                 - -
     Auxiliary Object Classes:    - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
     Mandatory Attributes:        - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
     Optional Attributes:         - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
                                  - - - - - - - - - - - - - -
```

The *operation* field will be one of the following depending on the operation performed:

Display OCT
Add OCT Entry
Delete OCT Entry
Modify OCT Entry

If *operation* is **Delete OCT Entry**, Mask 11 is displayed only with **Object Class Acronym**.

If *operation* is **Modify OCT Entry**, Mask 11 is displayed only with **Object Class Acronym**. If the object class acronym that is entered is an existing one, Mask 11 is displayed again with all values belonging to that OCT entry. Afterward, all fields can be modified except **Object Class Acronym** itself.

Mask 11 displays the following fields:

Object Class Acronym
Enter the object class acronym. No blanks are allowed in this acronym.

Object Class Name
Enter the object class name. No blanks are allowed in this name.

Acronyms of Superclasses
Enter up to 10 acronyms of object classes. These acronyms are separated by spaces. The object classes must exist in the OCT.

Object Identifier
Enter the object identifier as a printable string; for example, **85.6.12**.

Object Class Kind
Select one of the following valid object class kinds:

- **Abstract** (for example, **TOP, PER**)
- **Alias** (for example, **ALI**)
- **Auxiliary** (for example, **MUS**)
- **Structural** (for example, **ORP**)

File Number
Enter the number of the file in which the objects of this object class are stored. The number can be greater than 0 (0 is reserved for the schema object) or -1, which indicates that there are no concrete objects to be stored because this object class is only an abstract object class, alias object class, or auxiliary object class.

Auxiliary Object Classes
Enter a list of up to 10 acronyms of object classes. These acronyms are separated by spaces. They must exist as auxiliary object classes in the OCT.

Mandatory Attributes
Enter up to 10 acronyms of mandatory attributes. These must be separated by spaces.

Optional Attributes
Enter up to 50 acronyms of optional attributes. These must be separated by spaces.

Mask 11 uses the following function keys:

<Menu> Performs the selected operation (if the operation is **Display OCT**, displays the next OCT element).

<Return> Goes to the next field. If the field is the last one, performs the operation. (If the operation is **Display OCT**, displays the next OCT element.)

<Scroll Down>
Displays the next OCT element.

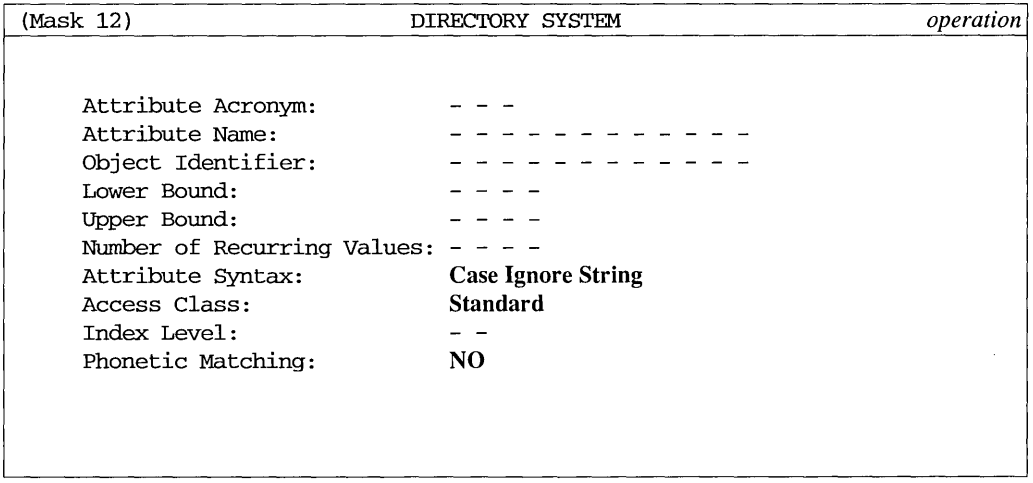
<Scroll Up>
Displays the previous OCT element.

<Del> Cancels the operation; returns to Mask 9.

9.1.7 Mask 12: AT Mask

Mask 12 (Figure 9-7) is used to display or enter the attributes in the AT.

Figure 9–7. Mask 12: AT Mask



The *operation* field will be one of the following depending on the operation being performed:

Display AT
Add AT Entry
Delete AT Entry
Modify AT Entry

If *operation* is **Delete AT Entry**, only **Attribute Acronym** is displayed in the mask.

If *operation* is **Modify AT Entry**, Mask 12 is displayed only with **Attribute Acronym**. If the attribute acronym entered is correct, Mask 12 is displayed again with all values belonging to that AT entry. Afterward, all fields can be modified except **Attribute Acronym** itself.

Mask 12 displays the following fields:

Attribute Acronym
Enter the attribute acronym. No spaces are allowed in the attribute acronym.

Attribute Name
Enter the attribute name. No spaces are allowed in the attribute name.

Object Identifier

Enter the object identifier as a printable string; for example, **85.4.6.**

Lower Bound

Enter the minimum length of the attribute value (1 to 9999).

Upper Bound

Enter the maximum length of the attribute value (1 to 9999).

Number of Recurring Values

Enter the maximum number of values an attribute can have as follows:

0	Unlimited number of values
1	One value (that is, the attribute is not a recurring attribute)
>1	Restricted number of values

If the number of recurring values is greater than 1, then a fixed amount of space is reserved. If the value is 0, no space is reserved; space is acquired dynamically depending on the number of values entered. Therefore, it is recommended that 0 be specified if there is a large number of possible values.

Attribute syntax

Select one of the following attribute syntaxes by pressing the space bar:

- Case Ignore String (default)
- Case Exact String
- Octet String
- ACL syntax
- Search Guide syntax
- Password syntax
- Country Name syntax
- Presentation Address
- Telephone Number syntax

- Fax Number syntax
- TTX-ID syntax
- Telex Number syntax
- Postal Address syntax
- Case Ignore List
- Boolean syntax
- Integer syntax
- UTC Time
- Numeric String
- Printable String
- Object Identifier syntax
- DN
- Preferred Delivery Method syntax
- Any syntax
- ASN.1 syntax
- MHS O/R Address syntax
- MHS O/R Name syntax
- MHS DL Submit Permission syntax
- MHS Preferred Delivery Method syntax

Access Class

Select one of the following access classes by pressing the space bar:

- **Public**
- **Standard** (default)
- **Sensitive** (defines the priority of an attribute for a search query; the range is between a very low priority of 0 and a very high priority of 99)

Index Level

Select the index level for the attribute. The index level defines the priority of an attribute for a search query; the range is between 0 (very low priority) and 99 (very high priority).

Phonetic Matching

Select one of the following phonetic matching flags by pressing the space bar:

YES The name of the attribute may vary slightly in search queries (for example, **Munchen/Muenchen**).

NO No phonetic matching (default).

Mask 12 uses the following function keys:

<Menu> Performs the selected operation. (If the operation is **Display AT**, displays the next AT element.)

<Return> Goes to the next field. If the field is the last one, performs the operation. (If the operation is **Display AT**, displays the next AT element.)

<Scroll Down>

Displays the next AT element.

<Scroll Up>

Displays the previous AT element.

<Del> Cancels the operation; returns to Mask 9.

9.2 Operations

Each of the following sections deals with one of the operations used for schema administration.

9.2.1 Store Schema

The Store Schema operation transfers the (modified) SRT, OCT, and AT from the memory of the administration program to the DSA.

After the new schema is successfully stored in the DSA, the DIT is reorganized according to the new schema; for example:

- If a structure rule is deleted from the SRT of the schema object, then any object of that structure rule is also deleted in the DIT.
- If an attribute is deleted from the OCT of the schema object, all the occurrences of that attribute are also deleted from all objects in the DIT.
- If the length of an attribute is shortened, the values of the attribute are also shortened.
- If the actual length of a naming attribute value is shortened, the object is deleted from the DIT.
- If the actual length of a structured attribute value like **Presentation-Address** or **Search-Guide** is shortened, the attribute value is deleted from the object.
- If an attribute with Distinguished Name syntax or an ACL contains the DN of an object whose name structure no longer conforms to the SRT, that DN is deleted from the attribute.

  If the ACL no longer contains the name of any object that may access the attributes of a particular access class, the object is deleted from the DIT.
- If all attribute values of a particular attribute are deleted, the attribute is deleted from the object.
- If an attribute is deleted from an object that is mandatory for a particular object class, the object will no longer belong to that object class. If it is the structural object class, the object will be removed from the DIT.
- If an optional attribute is made mandatory for a particular object class, then any object of that object class that is missing that attribute will no longer belong to that object class. If it is the structural object class of the object, the object is deleted from the DIT.
- If an object is deleted from the DIT, all of its subordinates are deleted also.

Mask Sequence

Mask 9 Select option number 1.

If the operation fails, the old schema is retained in the DSA and the new schema is stored locally in a file. The new schema can be loaded from the file back into the administration program memory with the Load Schema operation.

Note: The Store Schema operation updates the DSA's schema. The DSA process (**gdsdsa**) then changes the entire DIT and generates the **asn1_attr** file (in **/opt/dcelocal/var/directory/gds/dsa/dir***x*, where *x* is the directory ID), which is used for ASN.1 encoding and decoding. This file contains information on attribute types and their representation.

The C-stub process (**gdscstub**), which also performs ASN.1 encoding and decoding on the client side, uses an **asn1_attr** file in **/opt/dcelocal/var/adm/directory/gds/conf**. Because the C-stub process is a client component and the server does not know all its clients, the **asn1_attr** file is not updated after the schema is changed.

The C-stub's **asn1_attr** file contains information on attribute types and their representation of attributes that are handled by any DSAs the client ever wants to contact. The administrator is responsible for updating the **asn1_attr** file and must introduce at least all the complex attribute syntaxes. The information can be taken from the appropriate **asn1_attr** file used by the DSA. The administrator must also update the first line of the DSAs **asn1_attr** file with the total number of attributes stored in the **asn1_attr** file.

9.2.2 Load Schema

If a schema is stored in a file because a Store Schema operation failed, then the Load Schema operation transfers it back into the administration program memory.

Mask Sequence

Mask 9 Select option number 2.

9.2.3 Display SRT

The Display SRT operation displays the elements of the SRT.

Mask Sequence

Mask 9 Select option number 3.

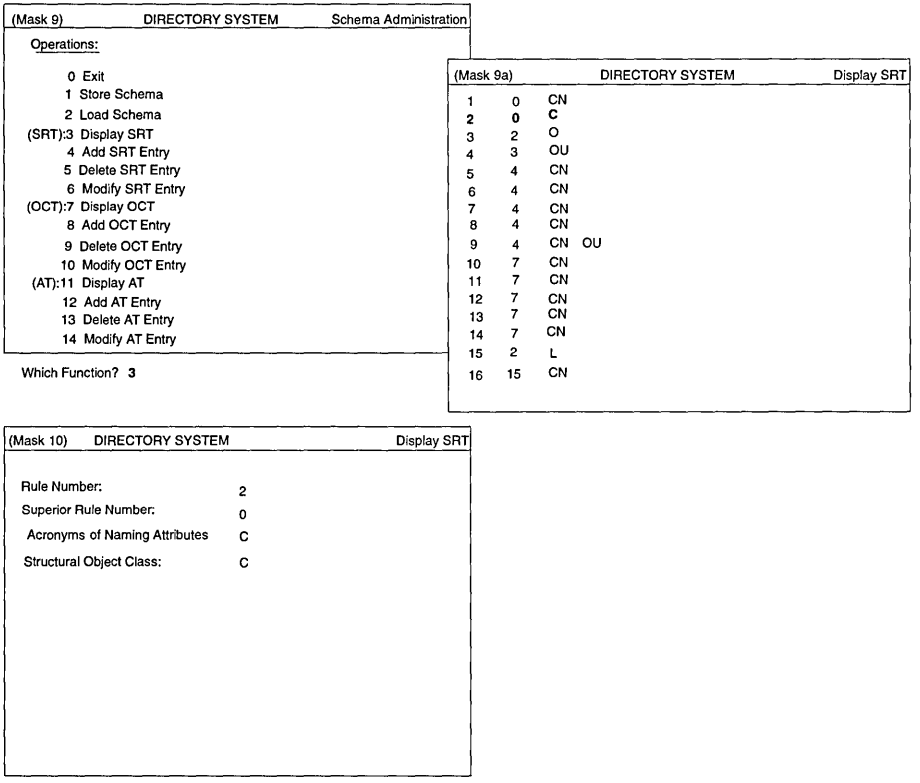
Mask 9a Select the element to be displayed.

Mask 10 Displays the element of the SRT.

Figure 9-8 shows a sample mask sequence that displays all the structure rules of the default schema (in Mask 9a) and that later displays one specific structure rule in Mask 10 (the structure rule that was selected in Mask 9a). The administrator's selections are highlighted in bold.

After the administrator selects the second item in Mask 9a (by pressing **<Return>** with the cursor on that item) and switches to the detailed information of that structure (by pressing **<F1>**), Mask 10 is displayed containing the information shown in the figure.

Figure 9–8. Sample Display SRT Operation



9.2.4 Add SRT Entry

Th Add SRT Entry operation adds a new structure rule to the SRT.

The operation adds the new structure rule in the memory only. The Store Schema operation must be called to add the structure rules to the DSA.

Mask sequence

Mask 9 Select option number 4.

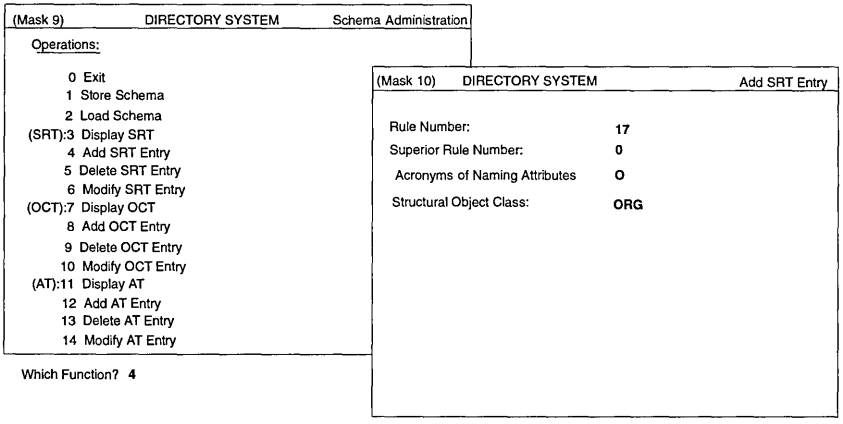
Mask 10 Enter the structure rule.

The following conditions must exist for this operation to complete successfully:

- The naming attributes must already exist in the AT, have Case Ignore String syntaxes, and must belong to an object class.
- The object class must exist as a structural object class in the OCT.
- The superior rule must exist in the SRT.

Figure 9-9 shows a sample mask sequence that creates a new structure rule, **17**, which represents an **Organization** under **ROOT**. The administrator's input is highlighted in bold.

Figure 9–9. Sample Add SRT Entry Operation



9.2.5 Delete SRT Entry

The Delete SRT Entry operation deletes a structure rule from the SRT.

The operation deletes the structure rule from memory only. The Store Schema operation must then be called to delete it from the DSA.

Mask Sequence

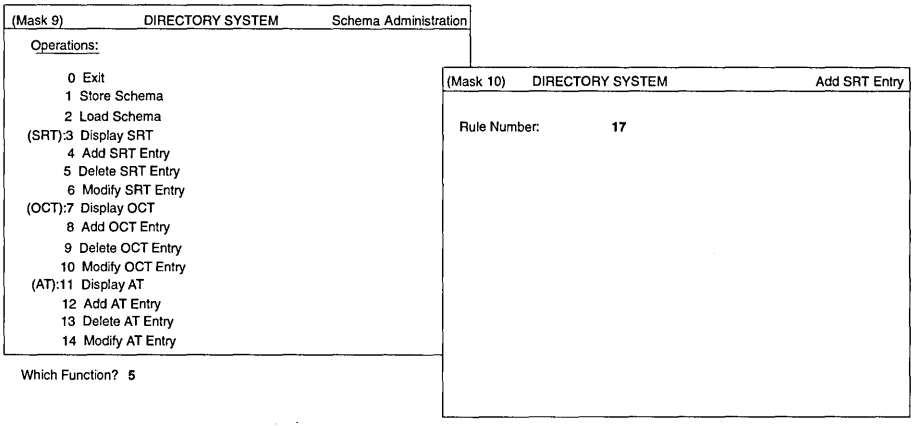
Mask 9 Select option number 5.

Mask 10 Enter the rule number of the structure rule to be deleted.

A structure rule can only be deleted if it is not one of the structure rules in the default schema or if it is not used as superior rule in any other structure rule.

Figure 9-10 shows a sample mask sequence that deletes structure rule 17. The administrator's input is highlighted in bold.

Figure 9–10. Sample Delete SRT Entry Operation



9.2.6 Modify SRT Entry

The Modify SRT operation modifies an existing structure rule in the SRT.

The operation modifies the structure rule in memory only. The Store Schema operation must then be called to modify the rule in the DSA.

Mask Sequence

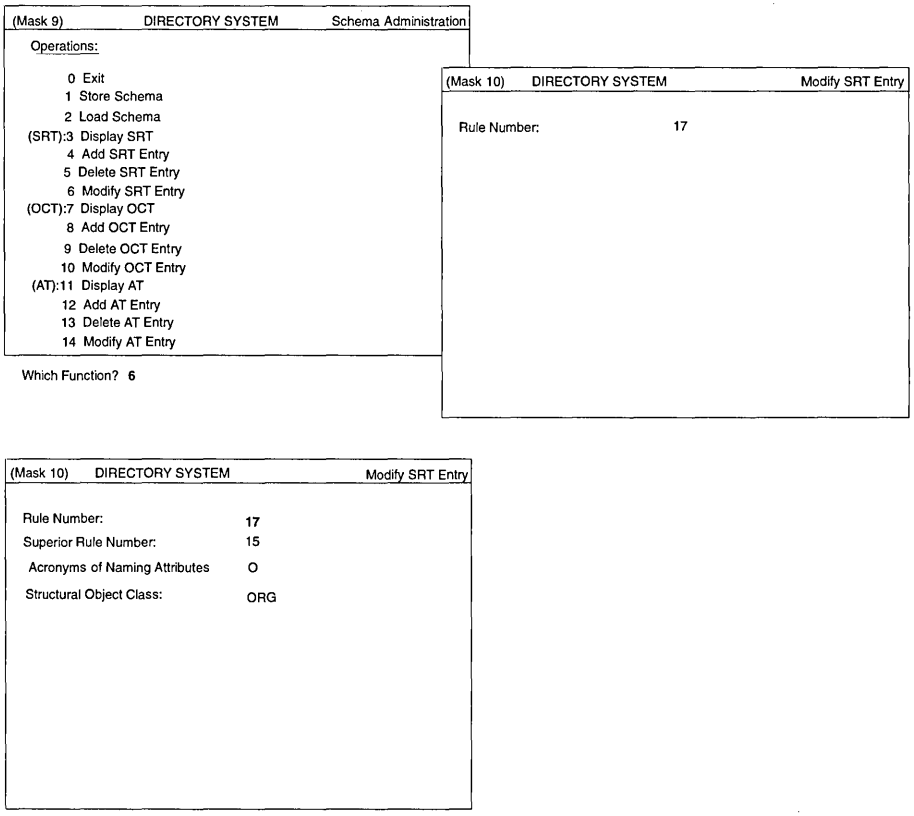
Mask 9 Select option number 6.

Mask 10 Only the **Rule Number** field is displayed initially. When a correct rule number is entered, Mask 10 is redisplayed with all values belonging to the SRT entry. Fields can then be modified.

Structure rules contained in the default schema cannot be modified.

Figure 9-11 shows a sample mask sequence that modifies structure rule 17 by assigning a new superior rule 15. The administrator's input is highlighted in bold.

Figure 9–11. Sample Modify SRT Entry Operation



The administrator enters option number 6 from Mask 9.

Mask 10 is displayed and the administrator enters the structure rule number of the rule to be modified. After the administrator presses **<Menu>** or **<Return>**, Mask 10 is displayed again with existing information about the rule to be modified. This information can then be overwritten with new information (for example, a new superior rule number).

9.2.7 Display OCT

The Display OCT operation displays the elements of the OCT.

Mask Sequence

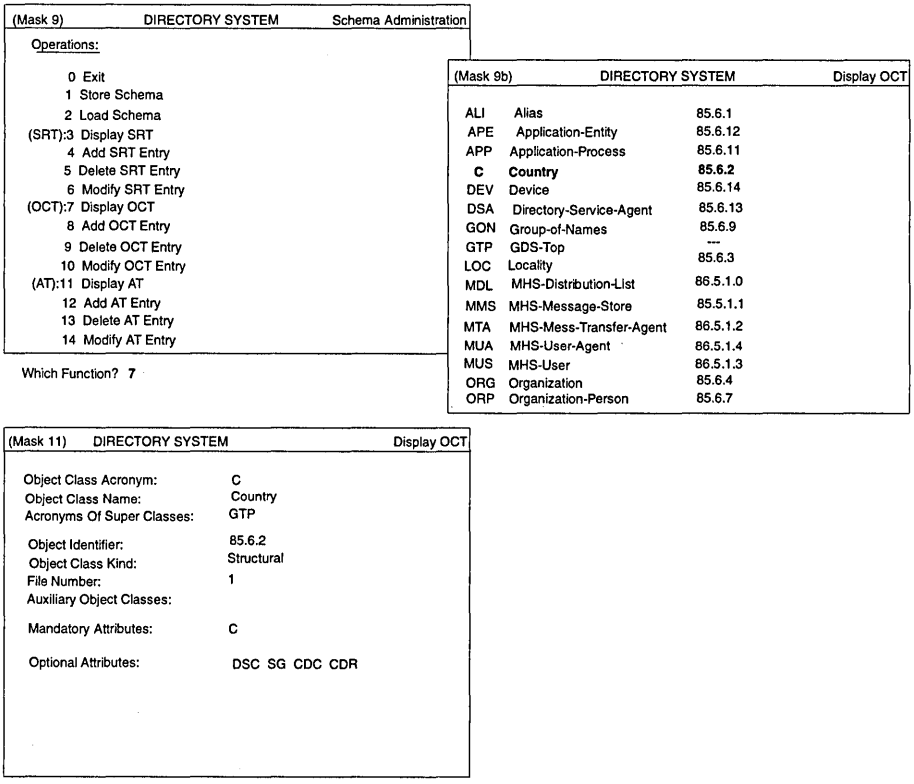
Mask 9 Select option number 7.

Mask 9b Select the object class to be displayed.

Mask 11 Displays the selected element of the OCT.

Figure 9-12 shows a sample mask sequence that displays all the object classes of the default schema in Mask 9b. Then in Mask 11, one specific object (selected in Mask 9b) is displayed. The administrator's selections are highlighted in bold.

Figure 9–12. Sample Display OCT Operation



9.2.8 Add OCT Entry

The Add OCT Entry operation adds a new object class in the OCT.

The operation adds the new object class in the memory only. The Store Schema operation must be called when adding objects in the DSA.

Mask sequence

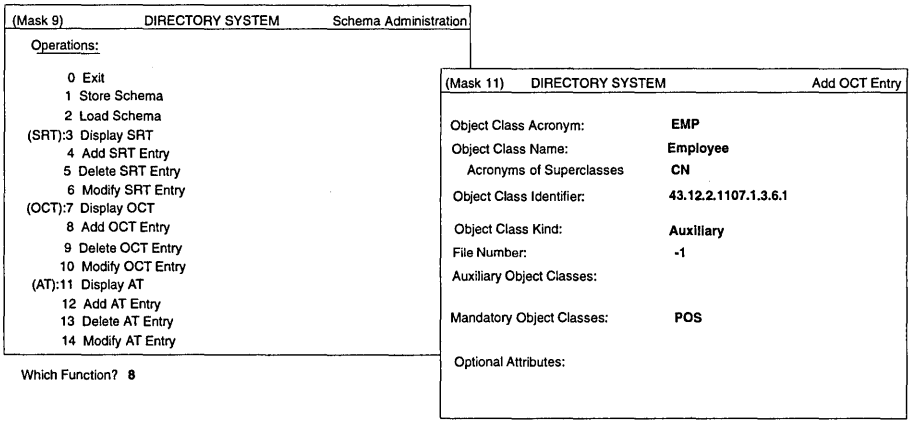
Mask 9 Select option number 8.

Mask 11 Enter the new object class information.

All attributes must already be in the AT, and the superclasses and auxiliary object classes must already be in the OCT.

Figure 9-13 shows a sample mask sequence that creates a new object class **Employee**. (Note that the mandatory attribute with acronym **POS** must already exist in the AT.) The administrator's input is highlighted in bold.

Figure 9–13. Sample Add OCT Entry Operation



9.2.9 Delete OCT Entry

The Delete OCT Entry operation deletes an object class from the OCT.

The operation deletes the object class from memory only. The Store Schema operation must then be called to delete it from the DSA.

Mask Sequence

Mask 9 Select option number 9.

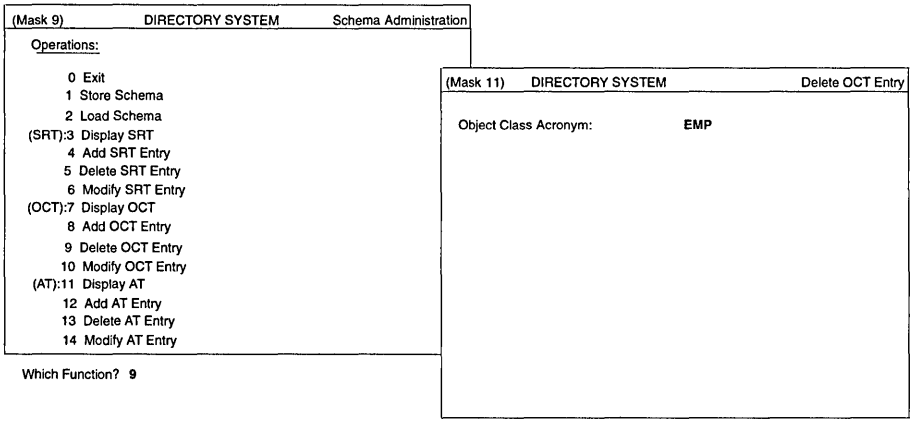
Mask 11 Select the object class to be deleted.

An object class can only be deleted if the following conditions exist:

- It has no subordinate object classes.
- The object class is not referenced in an SRT entry.
- It is not one of the standard object classes.
- It is not used as auxiliary object class.

Figure 9-14 shows a sample mask sequence that deletes the object class **Employee** by entering its acronym **EMP** in Mask 11.

Figure 9–14. Sample Delete OCT Entry Operation



9.2.10 Modify OCT Entry

The Modify OCT Entry operation modifies an existing object class in the OCT.

The operation modifies the object class entry in memory only. The Store Schema operation must then be called to change the entry in the DSA.

Mask Sequence

Mask 9 Select option number 10.

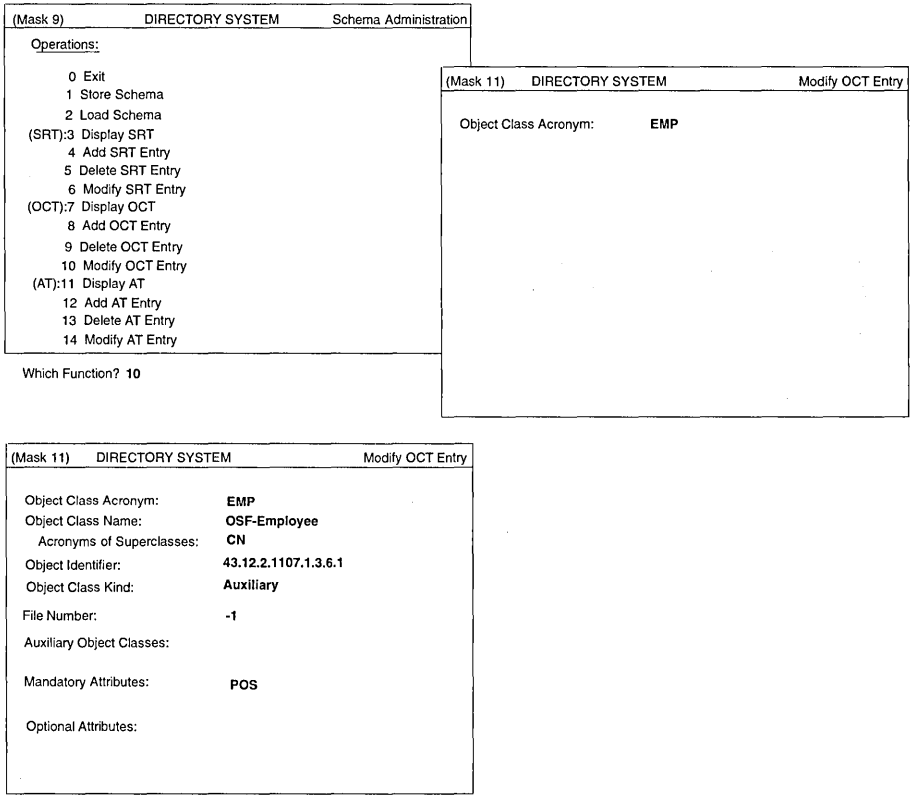
Mask 11 Only the **Object Class Acronym** field is displayed initially. When a correct object class acronym is entered, Mask 12 is redisplayed with all values belonging to the OCT entry. Fields can then be modified.

If the object class is contained in the default schema, then the following restrictions apply:

- The **Object Identifier**, **File Number**, **Superclasses**, and **Object Class Kind** fields cannot be modified.
- Mandatory attributes cannot be removed.

Figure 9-15 shows a sample mask sequence that modifies the object class **Employee** by entering its object class acronym, **EMP**, in Mask 11 and changing its values in Mask 11.

Figure 9–15. Sample Modify OCT Entry Operation



9.2.11 Display AT

The Display AT operation displays the elements of the AT.

Mask Sequence

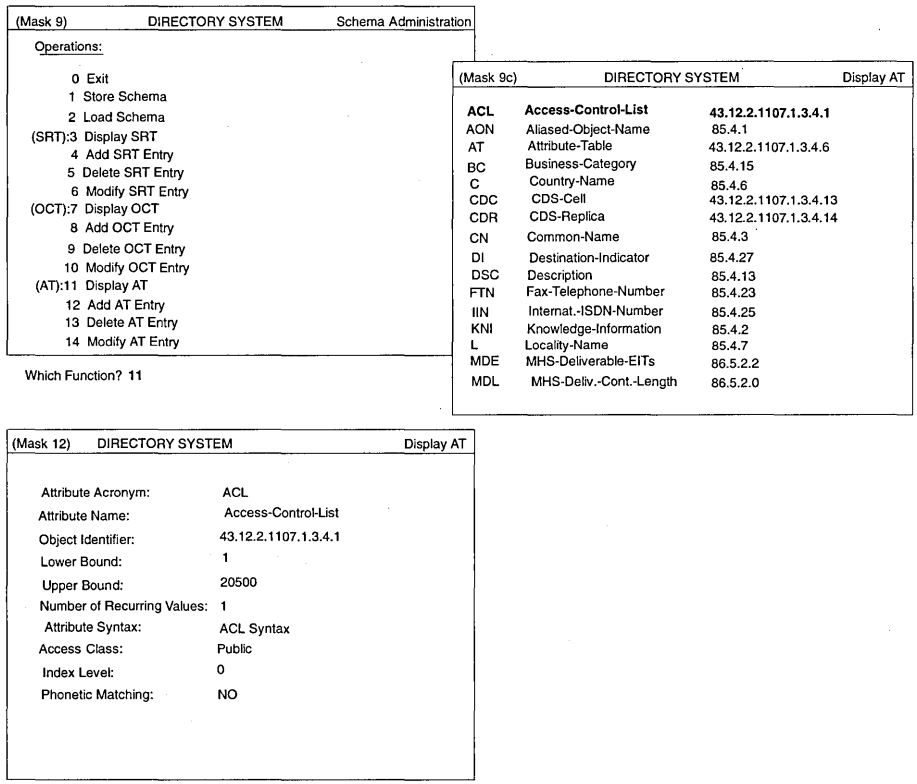
Mask 9 Select option number 11.

Mask 9c Select the attribute to be displayed.

Mask 12 Displays the selected element of the AT.

Figure 9-16 shows a sample mask sequence that displays all the attributes of the default schema in Mask 9c. A specific attribute is selected in Mask 9c and displayed in Mask 12. The administrator's selections are highlighted in bold.

Figure 9–16. Sample Display AT Operation



9.2.12 Add AT Entry

The Add AT Entry operation adds a new attribute in the AT.

The operation adds the new attribute in the memory only. The Store Schema operation must then be called for changes in the DSA.

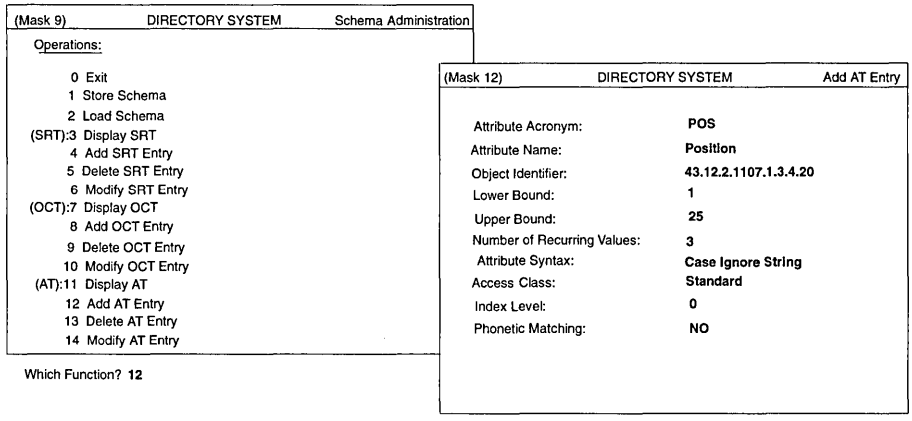
Mask Sequence

Mask 9 Select option number 12.

Mask 12 Enter the data for the new attribute.

Figure 9-17 shows a sample mask sequence that will create a new attribute **Position** with the information shown. The administrator's input is highlighted in bold.

Figure 9–17. Sample Add AT Entry Operation



9.2.13 Delete AT Entry

The Delete AT Entry operation deletes an attribute from the AT.

The operation deletes the attribute in the memory only. The Store Schema operation must then be called for changes in the DSA.

Mask Sequence

Mask 9 Select option number 13.

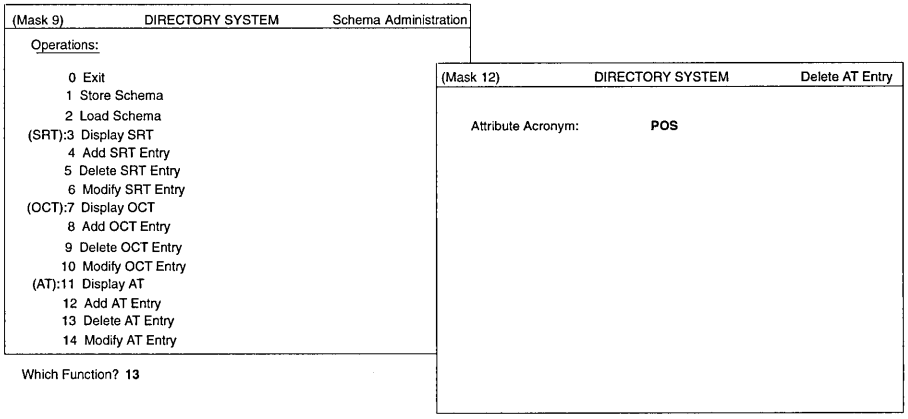
Mask 12 Enter the attribute acronym of the attribute to be deleted.

An attribute can only be deleted from the AT if the following conditions exist:

- It is not referenced as a naming attribute in the SRT.
- It is not referenced as a mandatory or an optional attribute in an OCT entry.
- It is not one of the standard attributes.

Fgure 10-18 shows a sample mask sequence that deletes the attribute **Position** by specifying its acronym, **POS**, in Mask 12. The administrator's input is highlighted in bold.

Figure 9–18. Sample Delete AT Entry Operation



9.2.14 Modify AT Entry

The Modify AT Entry operation modifies an existing attribute in the AT.

The operation modifies the attribute entry in memory only. The Store Schema operation must then be called to change the entry in the DSA.

Mask Sequence

Mask 9 Select option number 14.

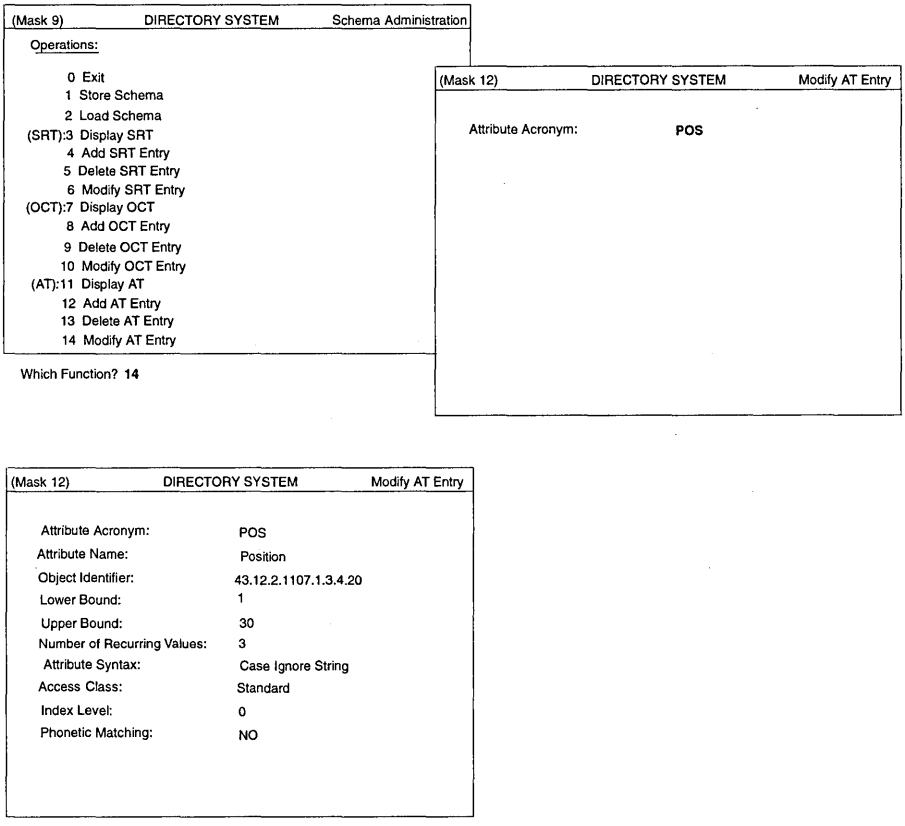
Mask 12 Only the **Attribute Acronym** field is displayed initially. When a correct attribute acronym is entered, Mask 12 is redisplayed with all values belonging to the AT entry. Fields can then be modified.

If the attribute is contained in the default schema, then its **Object Identifier** and **Attribute Syntax** fields cannot be modified.

If the attribute is used as a naming attribute in the SRT, its syntax cannot be changed. (Naming attributes always have **Case Ignore String** syntax.)

Figure 9-19 shows a sample mask sequence that modifies the upper bound of the attribute **Position** by entering its acronym, **POS**, in Mask 12. The administrator's input is highlighted in bold.

Figure 9–19. Sample Modify AT Entry Operation



Chapter 10

Shadow Administration

Shadow administration allows an administrator to manage shadows and shadowing jobs. An administrator can manage shadows and shadowing jobs only for those master object entries for which he or she is responsible. Shadow administration can be performed only when a bind has been made to the local DSA.

This chapter describes the shadow administration operations and interface. It shows all the masks that are used, their meanings, and input options. It also describes each of the shadow administration operations and the associated mask sequences.

Using shadow administration operations an administrator can

- Create shadows of master entries on remote DSAs
- Create shadowing jobs to update shadows
- Change the update frequency of a shadowing job
- Remove shadows and shadowing jobs
- Display existing shadowing jobs
- Display detailed information about errors that occurred during a shadow update

10.1 Creating Shadows of Master Entries on Remote DSAs

The unit of shadow creation on a target (remote) DSA can be either a single object or a subtree.

An administrator creates shadows on different DSAs to increase the availability of the data and reduce network access time and communication costs. A temporary inconsistency between master and shadow entries is tolerated by directory applications. If this inconsistency is not acceptable, only master information should be requested by a directory application.

10.2 Creating Shadowing Jobs

A shadowing job updates shadows of master entries. A shadowing job consists of the following:

- An object name and its interpretation (single object or subtree) for the object whose shadows should be updated
- The name of the target DSA where the shadows are present
- The update frequency that specifies how often the daemon process updating the shadows must be run

An administrator should select a high frequency for updates when master entries change frequently (for example, every 10 minutes) and the shadows must be updated as soon as possible. An administrator should select a low frequency for updates when the master entries change less frequently (for example, once a week). A medium frequency should be selected when the frequency of updates required falls somewhere between high and low.

Shadowing jobs can be created for shadows that have not been created by shadow administration. An inactive shadowing job can be created that can be activated later by the Update Shadowing Job operation. Shadowing is initiated by the master entries, so shadowing jobs only need to be created for subtrees or objects that are mastered by the bind DSA.

10.3 Removing Shadows and Shadowing Jobs

Shadows can be removed from remote DSAs when they are no longer required. The associated shadowing job is automatically removed from the local DSA.

A shadowing job can be removed without removing the shadows. This should only be done by an administrator when shadows have been removed by some other means such as subtree administration operations.

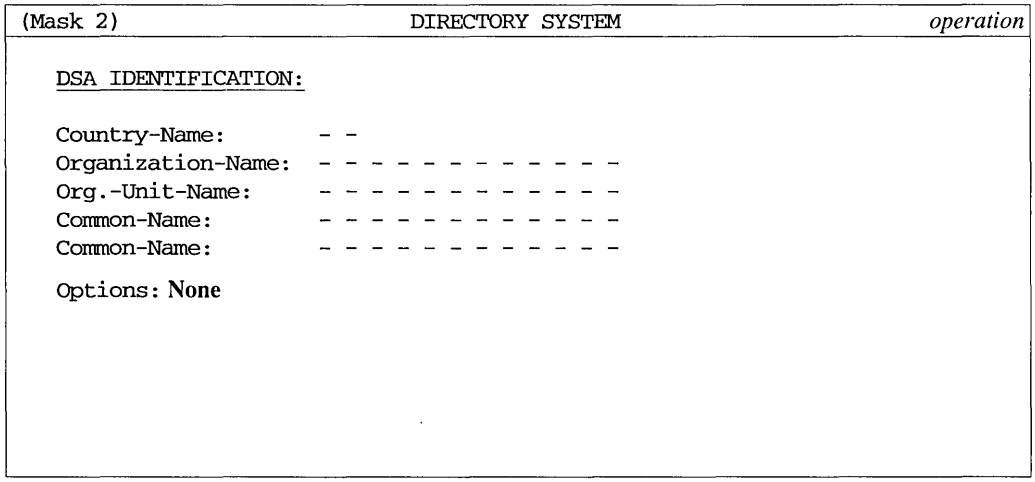
10.4 Displaying Shadowing Jobs and Error Information

An administrator can display the list of existing shadowing jobs using Masks 14g through 14j and can display errors that occurred during shadow updates using Mask 15.

10.5 Mask 2: DSA Identification

Mask 2 (Figure 10-1) is used to specify the DSA in which the shadows are to be managed.

Figure 10–1. Mask 2: DSA Identification



In Mask 2, the *operation* field will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Update Shadowing Job

Mask 2 displays the following fields:

Country-Name
Enter the **Country-Name** part of the DSAs DN.

Organization-Name
Enter the **Organization-Name** part of the DSAs DN.

Org.-Unit-Name
Enter the **Org.-Unit-Name** part of the DSAs DN.

Common-Name
Enter the **Common-Name** part of the DSAs DN.

Common-Name
Enter the **Common-Name** part of the DSAs DN (the DN of DSA to be selected).

Options Select one of the following options by pressing the space bar:

- **None** (default)
- **Changing Name Structure**

If **Changing Name Structure** is selected, all structure rules are displayed in Mask 5. Only structure rules that represent a DSA can be selected; that is, structure rules that have the **Presentation-Address** attribute.

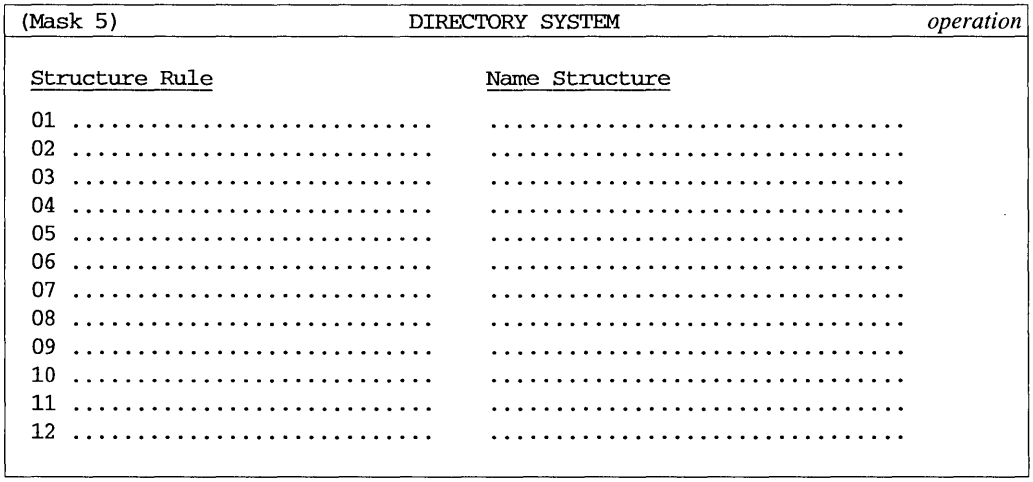
The only structure in the default schema that represents a DSA is structure rule 7.

After the selection is made, Mask 2 displays the selected name structure for entering the DN. This structure rule is also used for the next Logon to a Specific DSA operation.

10.5.1 Mask 5: Structure Rule

Mask 5 (Figure 10-2) is used to select the structure rule to be processed. The naming attribute of all structure rules stored in the SRT are displayed here. If the SRT contains more than 12 structure rules, **<Scroll Up>** and **<Scroll Down>** can be used to page through the mask.

Figure 10–2. Mask 5: Structure Rule




```
Which Structure Rule ?
```

The *operation* field in Mask 5 will be one of the following values depending on which operation is being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Update Shadowing Job

Mask 5 displays the following fields:

Which Structure Rule ?

Enter the number of the structure rule to be processed. Figure 10-3 shows an example of Mask 5 with a set of sample structure rule values.

Figure 10–3. Mask 5: Sample Structure Rules

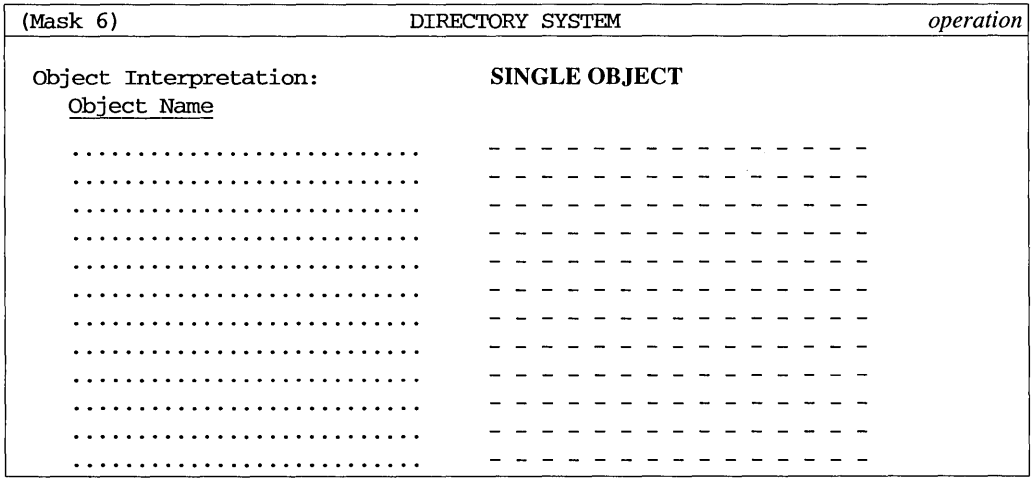
```
(Mask 5)                    DIRECTORY SYSTEM                    operation

Structure Rule                          Name Structure

01 Common-Name                          01
02 County-Name                          02
03 Organization-Name                    02-03
04 Org.-Unit-Name                       02-03-04
05 Common Name                          02-03-04-05
06 Common Name                          02-03-04-06
     Org.-Unit-Name
07 Common Name                          02-03-04-05-07
08 Locality-Name                        02-08
09 Common-Name                          02-08-09
10 Common-Name                          02-08-10
     Street-Address

Which Structure Rule ?
```

If the administration function **Shadow Administration** is selected in Mask 3 (see Figure 7-3), the system also displays the structure rule **00 ROOT** in Mask 5. The **00 ROOT** structure rule is only displayed if the object/subtree to be shadowed is to be entered. It is not shown if Mask 5 is used to specify the structure rule of the target DSA. If this structure rule is selected, all master entries can be managed as shadows in a remote DSA.

10.5.2 Mask 6: Object Name

Mask 6 (Figure 10-4) is used to input the object name or subtree name for shadow administration. Wildcards are not permitted when defining subtrees or the name of the target DSA.

Figure 10–4. Mask 6: Object Name



The *operation* field will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Update Shadowing Job

The **Object Interpretation** field is not displayed if the name of the target DSA needs to be entered.

Mask 6 displays the following fields:

Object Interpretation
Select one of the following object interpretations by pressing the space bar:

- **SINGLE OBJECT**
- **OBJECT AND ITS SUBORDINATES**

10.5.3 Mask 13: Shadow Operations

Mask 13 (Figure 10-5) is used to select the shadow administration operation to be executed.

Figure 10–5. Mask 13: Shadow Operations

```
(Mask 13)                    DIRECTORY SYSTEM          Shadow Administration

  OPERATIONS:

    0 Exit
    1 Create Shadows and Shadowing Job
    2 Create Shadowing Job
    3 Remove Shadows and Shadowing Job
    4 Remove Shadowing Job
    5 Update Shadowing Job
    6 Display Shadowing Jobs
    7 Display Update Errors
    8 Remove Update Error

  Which operation ?
```

Mask 13 displays the following field:

Which operation ?
Enter the number of the operation to be executed (0 to 8).

10.5.4 Mask 14a: Shadowing Job (Job State)

Mask 14a (Figure 10-6) is used to define the job state of the shadowing job.

If the shadowing job to be managed has already been selected with **Display Shadowing Jobs** option on Mask 13, Mask 14a overlays the previously displayed Mask 14g, 14h, 14i or 14j.

Figure 10–6. Mask 14a: Shadowing Job (Job State)

```
(Mask 14)                    DIRECTORY SYSTEM                    operation

Object Name:                 ....................................
                             ....................................

Object Interpretation:       interpretation

Target DSA:                  ....................................
                             ....................................

Job:                         ACTIVATE UPDATES
```

In Mask 14a, *operation* will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Update Shadowing Job
Cache Update

In the **Object Interpretation** field, *interpretation* will be one of the following values, depending on the object interpretation selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

Mask 14 displays the following fields:

Object Name
Enter the object name on which the operation will be performed.

Target DSA Enter the DSA on which the shadow job operation will be performed.

Job If *operation* is **Create Shadows and Shadowing Job**, or **Create Shadowing Job**, select one of the following values by pressing the space bar:

- **ACTIVATE UPDATES**
- **NO UPDATES**

If *operation* is **Update Shadowing** and an active job is displayed, select one of the following values by pressing the space bar:

- **ACTIVATE UPDATES IMMEDIATELY**
- **DEACTIVATE UPDATES**
- **MODIFY UPDATE FREQUENCY**

If *operation* is **Update Shadowing** and an inactive job is displayed, select one of the following values by pressing the space bar:

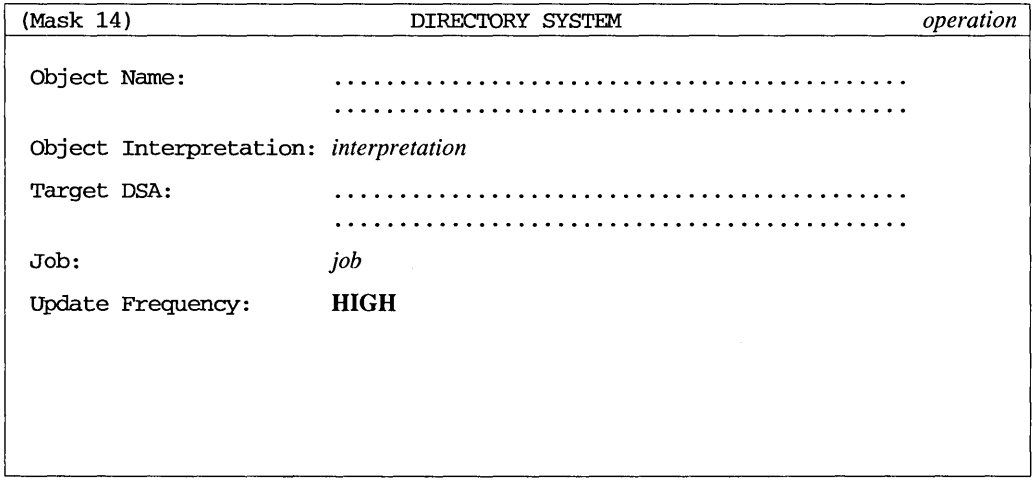
- **ACTIVATE UPDATES IMMEDIATELY**
- **ACTIVATE UPDATES**

If *operation* is **Cache Update**, the **Object Name**, **Object Interpretation**, and **Target DSA** fields are not displayed.

10.5.4.1 Mask 14b: Shadowing Job (Selection of Update Frequency)

Mask 14b (Figure 10-7) overlays Mask 14a (Figure 10-6), and is used to define the update frequency. It is displayed when the administration function **ACTIVATE UPDATES** or **MODIFY UPDATE FREQUENCY** is selected in the **Job** field in Mask 14a.

Figure 10–7. Mask 14b: Shadowing Job (Selection of Update Frequency)



In Mask 14b, *operation* will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Update Shadowing Job
Cache Update

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Job** field, *job* will be one of the following values depending on the value you selected in Mask 14a:

ACTIVATE UPDATES
MODIFY UPDATE FREQUENCY

Mask 14b displays the following fields:

Update Frequency

Select one of the following values by pressing the space bar:

- **HIGH**
- **MEDIUM**
- **LOW**

If **HIGH** is selected, the update job is executed every few minutes. This is the default for **Create Shadowing Job** or **Create Shadows and Shadowing Job.**

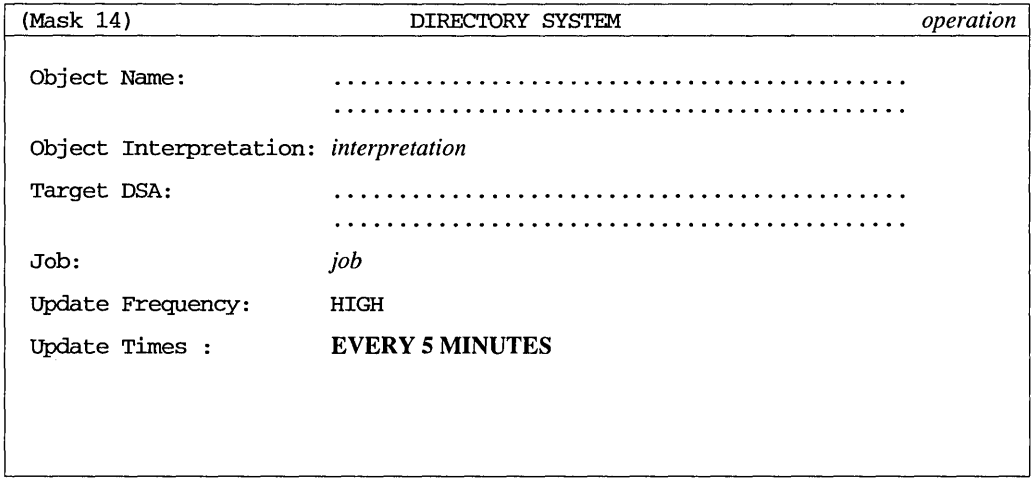
If **MEDIUM** is selected, the update job is executed every few hours.

If **LOW** is selected, the update job is executed every few days.

10.5.4.2 Mask 14c: Update Times if the Frequency is HIGH

Mask 14c (Figure 10-8) overlays Mask 14b (Figure 10-7) and is used to input the frequency in minutes if the update frequency selected is **HIGH.**

Figure 10–8. Mask 14c: Update Times if the Frequency is HIGH



In Mask 14c, *operation* will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Update Shadowing Job
Cache Update

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Job** field, *job* will be one of the following values depending on the value you selected in Mask 14a:

ACTIVATE UPDATES
MODIFY UPDATE FREQUENCY

Mask 14c displays the following field:

Update Times

Select one of the following update times by pressing the space bar:

- **EVERY 5 MINUTES**
- **EVERY 10 MINUTES**
- **EVERY 15 MINUTES**
- **EVERY 30 MINUTES**

The update time in minutes is relative to 0 minutes past the hour. For example, **EVERY 10 MINUTES** means that the updates are done at 0 minutes past the hour, 10 minutes past the hour, 20 minutes past the hour, 30 minutes past the hour, and so on.

10.5.4.3 Mask 14d: Update Times if the Frequency is MEDIUM

Mask 14d (Figure 10-9) overlays Mask 14b (Figure 10-7) and is used to input the frequency in hours if the update frequency is **MEDIUM**.

Figure 10–9. Mask 14d: Update Times if the Frequency is MEDIUM

```
(Mask 14)                    DIRECTORY SYSTEM                    operation

Object Name:                 ........................................
                             ........................................

Object Interpretation:       interpretation

Target DSA:                  ........................................
                             ........................................

Job:                         job

Update Frequency:            MEDIUM

Update Times :               EVERY HOUR
```

In Mask 14d, *operation* will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Update Shadowing Job
Cache Update

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Job** field, *job* will be one of the following values depending on the value you selected in Mask 14a:

ACTIVATE UPDATES
MODIFY UPDATE FREQUENCY

Mask 14d displays the following field:

Update Times

Select one of the following update times by pressing the space bar:

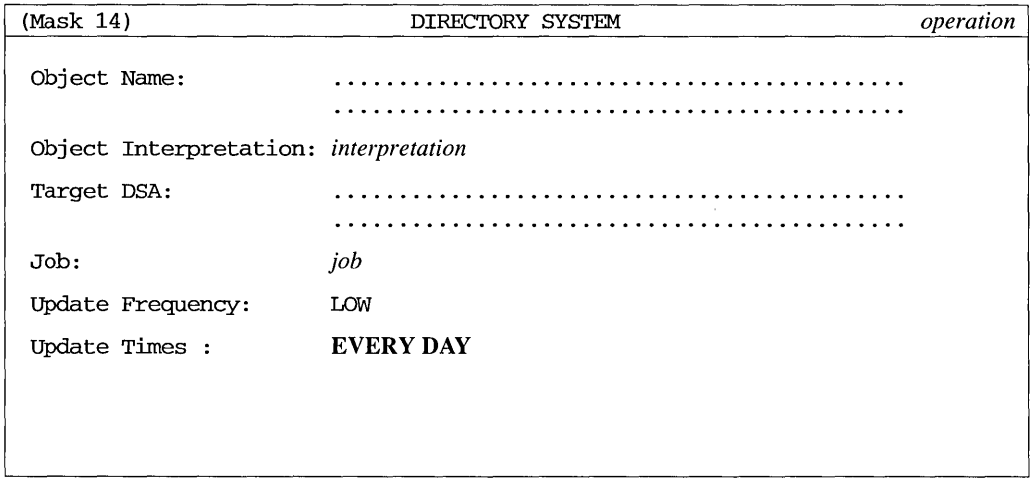
- **EVERY HOUR**
- **EVERY 2 HOURS**
- **EVERY 4 HOURS**
- **EVERY 6 HOURS**
- **EVERY 12 HOURS**

The update times in hours are relative to 0 a.m. For example, **EVERY 2 HOURS** means that the updates are carried out at 0 a.m., 2 a.m., 4 a.m., 6 a.m., and so on, up to 10 p.m.

10.5.4.4 Mask 14e: Update Times if the Frequency is LOW

Mask 14e (Figure 10-10) overlays Mask 14b (Figure 10-7) and is used to input the frequency in hours if the update frequency is **LOW**.

Figure 10–10. Mask 14e: Update Times if the Frequency is LOW



In Mask 14e, *operation* will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Update Shadowing Job
Cache Update

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Job** field, *job* will be one of the following values depending on the value you selected in Mask 14a:

ACTIVATE UPDATES
MODIFY UPDATE FREQUENCY

Mask 14e displays the following field:

Update Times

Select one of the following update times by pressing the space bar:

- **EVERY DAY**
- **ONCE A WEEK**
- **TWICE A WEEK**

10.5.4.5 Mask 14f: Days and Hours if the Frequency is LOW

Mask 14f (Figure 10-11) overlays Mask 14b (Figure 10-7) and is used to input the days and hours if the update frequency is **LOW**.

Figure 10–11. Mask 14f: Days and Hours if the Frequency is LOW

```
(Mask 14)                    DIRECTORY SYSTEM                    operation

Object Name:                 ........................................
                             ........................................
Object Interpretation:       interpretation
Target DSA:                  ........................................
                             ........................................
Job:                         job
Update Frequency:            LOW
Update Times:                times
  Day of Week:               SUNDAY
  Hours[0-23]:               - -
  Day of Week:               SUNDAY
  Hours[0-23]:               - -
```

In Mask 14f, *operation* will be one of the following values depending on the operation being performed:

Create Shadows and Shadowing Job
Create Shadowing Job
Update Shadowing Job
Cache Update

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Job** field, *job* will be one of the following values depending on the value you selected in Mask 14a:

ACTIVATE UPDATES
MODIFY UPDATE FREQUENCY

In the **Update Times** field, *times* will be one of the following values depending on the value you selected in Mask 14c, 14d, or 14e:

EVERY DAY
ONCE A WEEK
TWICE A WEEK

If *times* is **EVERY DAY**, the **Day of Week** fields are not displayed, and there is only one **Hours** field (directly under **Update Times**).

If *times* is **ONCE A WEEK**, there is only one **Day of Week** field and one **Hours** field.

If *times* is **TWICE A WEEK**, then two **Day of Week** fields and two **Hours** fields are displayed (see the preceding mask).

Mask 14f displays the following fields:

Day of Week

Select one of the following by pressing the space bar:

- **MONDAY**
- **TUESDAY**
- **WEDNESDAY**

- **THURSDAY**
- **FRIDAY**
- **SATURDAY**
- **SUNDAY**

Hours Specify the hour of the update.

10.5.4.6 Mask 14g: Display an Active Shadowing Job with HIGH Frequency

Mask 14g (Figure 10-12) is used to display an active job with the update frequency set to **HIGH**.

Figure 10–12. Mask 14g: Active Shadowing Job with HIGH Frequency

```
(Mask 14)                    DIRECTORY SYSTEM                    operation

Object Name:                 ........................................
                             ........................................
Object Interpretation:       interpretation
Target DSA:                  ........................................
                             ........................................
Job:                         ACTIVE
Update Frequency:            HIGH
Update Times:                times
```

In Mask 14g, *operation* will be one of the following values depending on the operation being performed:

Update Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Display Shadowing Jobs

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Job** field, *job* will be one of the following values depending on the value you selected in Mask 14a:

ACTIVATE UPDATES
MODIFY UPDATE FREQUENCY

In the **Update Times** field, *times* will be one of the following values depending on the value you selected in Mask 14d:

EVERY 5 MINUTES
EVERY 10 MINUTES
EVERY 15 MINUTES
EVERY 30 MINUTES

If *operation* is **Update Shadowing Job**, this mask is overlaid by a series of masks ranging between 14a and 14f, depending on what is selected in these masks.

10.5.4.7 Mask 14h: Display an Active Shadowing Job with MEDIUM Frequency

Mask 14h (Figure 10-13) is used to display an active shadowing job with the update frequency set to **MEDIUM**.

Figure 10–13. Mask 14h: Active Shadowing Job with MEDIUM Frequency

```
(Mask 14)                    DIRECTORY SYSTEM                    operation

Object Name:                 ........................................
                             ........................................

Object Interpretation:       interpretation

Target DSA:                  ........................................
                             ........................................

Job:                         ACTIVE

Update Frequency:            MEDIUM

Update Times:                times
```

In Mask 14h, *operation* will be one of the following values depending on the operation being performed:

Update Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Display Shadowing Jobs

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Update Times** field, *times* will be one of the following values depending on the value you selected in Mask 14d:

EVERY HOUR
EVERY 2 HOURS
EVERY 4 HOURS
EVERY 6 HOURS
EVERY 12 HOURS

If *operation* is **Update Shadowing Job**, this mask is overlaid by a series of masks ranging between 14a and 14f, depending on what is selected in these masks.

10.5.4.8 Mask 14i: Display an Active Shadowing Job with LOW Frequency

Mask 14i (Figure 10-14) is used to display an active shadowing job with the update frequency set to **LOW**.

Figure 10–14. Mask 14i: Active Shadowing Job with LOW Frequency

(Mask 14)	DIRECTORY SYSTEM	*operation*
Object Name:	.....	
	.....	
Object Interpretation:	*interpretation*	
Target DSA:	.....	
	.....	
Job:	ACTIVE	
Update Frequency:	LOW	
Update Times:	*times*	
Day of Week:	*day of week*	
Hours[0-23]:	*hours*	
Day of Week:	*day of week*	
Hours[0-23]:	*hours*	

In Mask 14i, *operation* will be one of the following values depending on the operation being performed:

Update Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Display Shadowing Jobs

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

In the **Update Times** field, *times* will be one of the following values depending on the value you selected in Mask 14e:

EVERY DAY
ONCE A WEEK
TWICE A WEEK

In the **Day of the Week** field, *day of the week* will be one of the following values depending on the value you selected in Mask 14f:

MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY

In the **Hours** field, *hours* will be the value between 0 and 23 you specified in Mask 14f.

If *times* is **EVERY DAY**, the **Day of Week** fields are not displayed, and there is only one **Hours** field (directly under **Update Times**).

If *times* is **ONCE A WEEK**, there is only one **Day of Week** field and one **Hours** field.

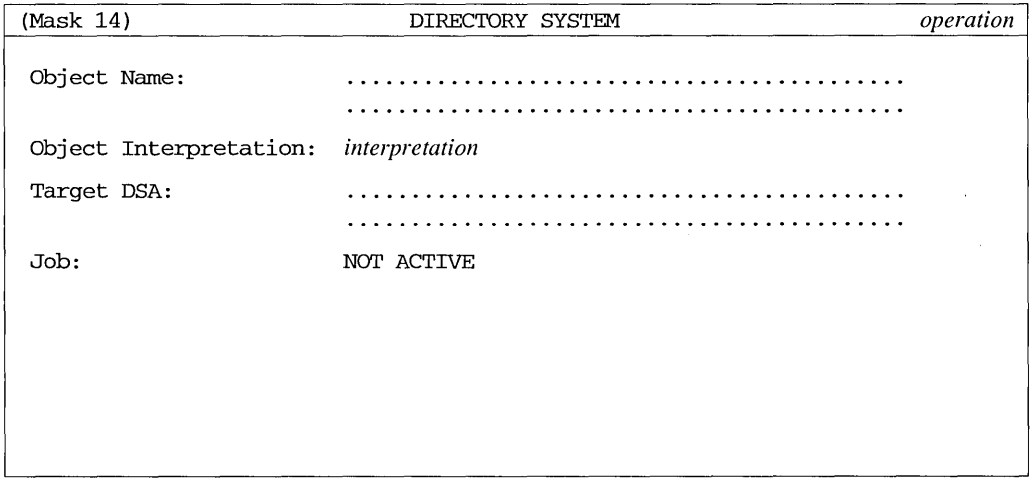
If *times* is **TWICE A WEEK**, two **Day of Week** fields and two **Hours** fields are displayed (see Figure 10-11).

If *operation* is **Update Shadowing Job**, this mask is overlaid by a series of masks ranging between 14a and 14f, depending on what is selected in these masks.

10.5.4.9 Mask 14j: Display an Inactive Shadowing Job

Mask 14j (Figure 10-15) is used to display an inactive shadowing job.

Figure 10–15. Mask 14j: Inactive Shadowing Job



In Mask 14j, *operation* will be one of the following values depending on the operation being performed:

Update Shadowing Job
Remove Shadows and Shadowing Job
Remove Shadowing Job
Display Shadowing Jobs

In the **Object Interpretation** field, *interpretation* will be one of the following values depending on the object interpretation you selected in Mask 6:

SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

If *operation* is **Update Shadowing Job**, this mask is overlaid by a series of masks ranging between 14a and 14f, depending on what is selected in these masks.

10.5.4.10 Mask 15: Error Mask

Mask 15 (Figure 10-16) is either used to display update errors or to specify which operations are not successful and must be called again by the delta update daemon when it is next activated.

Figure 10–16. Mask 15: Error Mask

```
(Mask 15)                    DIRECTORY SYSTEM                    operation

Object Name:          ........................................
                      ........................................

Target DSA:           ........................................
                      ........................................

Operation:            op_code
Error Time:           yy-mm-dd/hh:mm
Error Count:          count
Error Code:           code

Remove Update Error: NO
```

In Mask 15, *operation* will be one of the following values depending on the operation being performed:

Display Update Errors
Remove Update Error

In the **Operation** field, *op_code* will be one of the following values depending on the type of unsuccessful operation:

ADD
MODIFY
REMOVE
MODIFY RDN

Mask 15 displays the following fields:

Error Time
The time the error occurred in the following format: *yy-mm-dd/hh:mm* (year-month-day/hour:minute).

Error Count
The number of erroneous attempts to perform the shadow update.

Error Code
The reason for the error.

If *operation* is **Remove Update Error**, the following field is displayed:

Remove Update Error
Select one of the following by pressing the space bar:

YES	The update error is removed and the update is not repeated by the delta daemon process.
NO	The update error is not removed and the update is repeated (default).

Mask 15 uses the following function keys:

<Menu>	Displays the next update error. The erroneous update operation is not repeated if the operation is **Remove Update Error** and **Remove Update Error** is set to **YES**.
<Return>	Displays the next update error. The erroneous update operation is not repeated if the operation is **Remove Update Error** and **Remove Update Error** is set to **YES**.
<Scroll Down>	Displays the next update error. The erroneous update operation is not repeated if the operation is **Remove Update Error** and **Remove Update Error** is set to **YES**.
<Del>	Cancels the operation; returns to Mask 13.

10.6 Operations

Each of the following sections deals with one of the operations used for shadow administration.

10.6.1 Cache Update

The Cache Update operation displays, activates, deactivates, or changes the update frequency of the **Cache Update** job. This operation is only available after a **Logon to the DUA Cache** operation is performed.

Mask sequence

Mask 1: Select the **Logon to the DUA cache** option (see Chapter 7).

Mask 3: Select option number 2 (**Cache Update**; see Chapter 7). Mask 14g, 14h, 14i, or 14j is then displayed.

Mask 14a: Select the job administration function from the **Job** field. If you select **ACTIVATE UPDATES** or **MODIFY UPDATE TIMES**, then Mask 14b is displayed.

Mask 14b: Select the update frequency.

Depending on the format of the update times, Mask 14c, 14d, or 14e (followed by Mask 14f) is displayed.

Mask 14c Enter the frequency in minutes.

Mask 14d Enter the frequency in hours.

Mask 14e Enter the frequency in days per week.

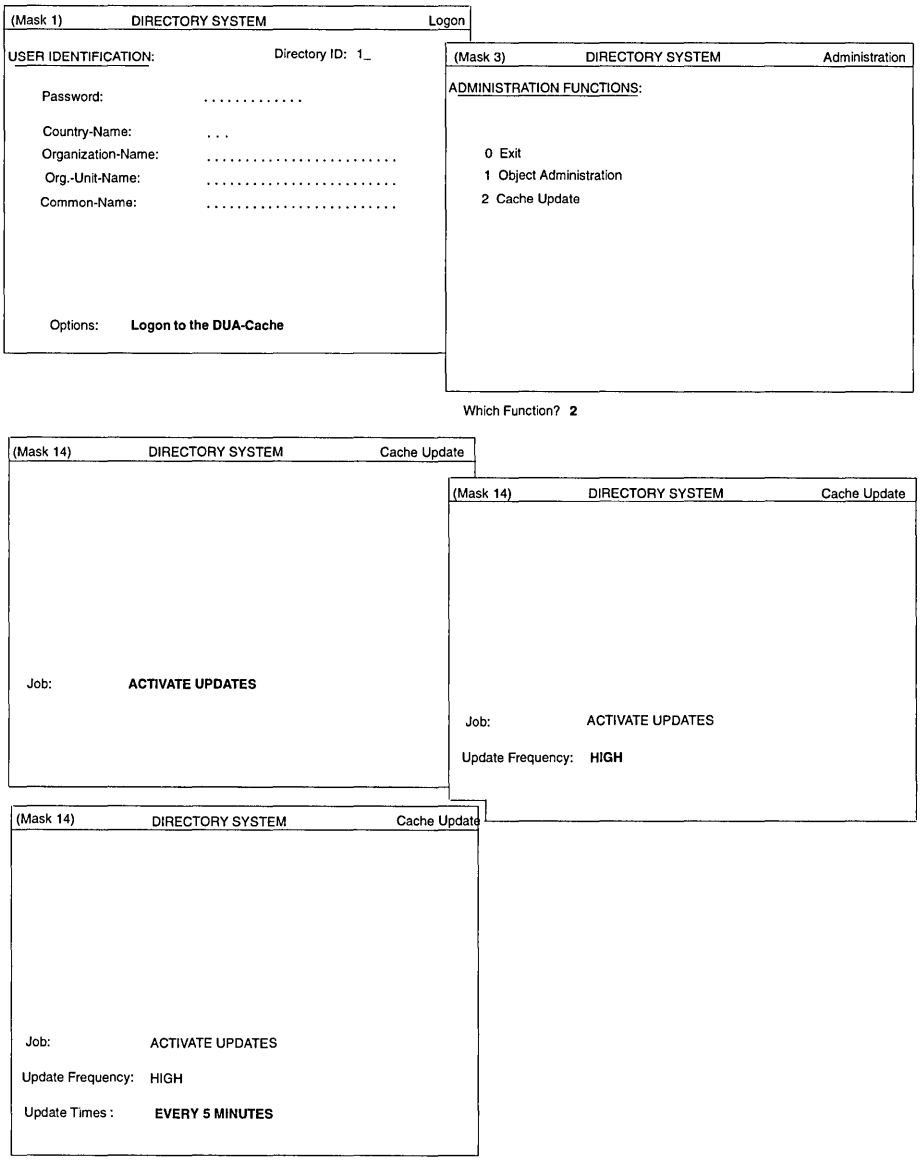
Mask 14f Enter the frequency in a day of the week (and in an hour of that day).

Note: There is one inactive **Cache Update** job for each configured directory after configuration is completed.

The **Cache Update** process updates every object stored in the cache by using the master information stored on all DSAs.

Figure 10-17 shows a sample mask sequence that changes the cache update frequency for the cache to once every five minutes.

Figure 10–17. Sample Cache Update Operation



10.6.2 Create Shadows and Shadowing Job

The Create Shadows and Shadowing Job operation generates an active or inactive shadowing job and creates the shadow entries of the object or subtree in the specified DSA or DSAs. An inactive shadowing job is created if **NO UPDATES** is selected in Mask 14a.

The creation of a new job is rejected if another job (of which the new job is a subset) already exists for the same target DSA. For example, a job cannot be created for the subtree **/C=US/0=Smith Ltd** when a job **/C=US** for the whole subtree already exists for the same DSA.

Existing jobs are removed for the same target DSA if these jobs are a subset of the new job. For example, the job **/C=US/O=Smith Ltd.** is removed when a job **/C=US** is created for a whole subtree.

Mask sequence

Mask 3	Select option number 3 (see Chapter 7).
Mask 13	Select option number 1.
Mask 5	Select the structure rule of the object or the root of the subtree whose master entries are to be copied.
Mask 6	Enter the object name of the object or the root of the subtree whose master entries are to be copied.
Mask 2	Enter the DN of target DSA in which the shadow entries are to be created.
Mask 14a	Select the job administration function from the **Jobs** field. If you select **ACTIVATE UPDATES**, then Mask 14b is displayed.
Mask 14b	Select the update frequency. Depending on the format of the update times, Mask 14c, 14d, or 14e (followed by Mask 14f) is displayed.
Mask 14c	Enter the frequency in minutes.
Mask 14d	Enter the frequency in hours.

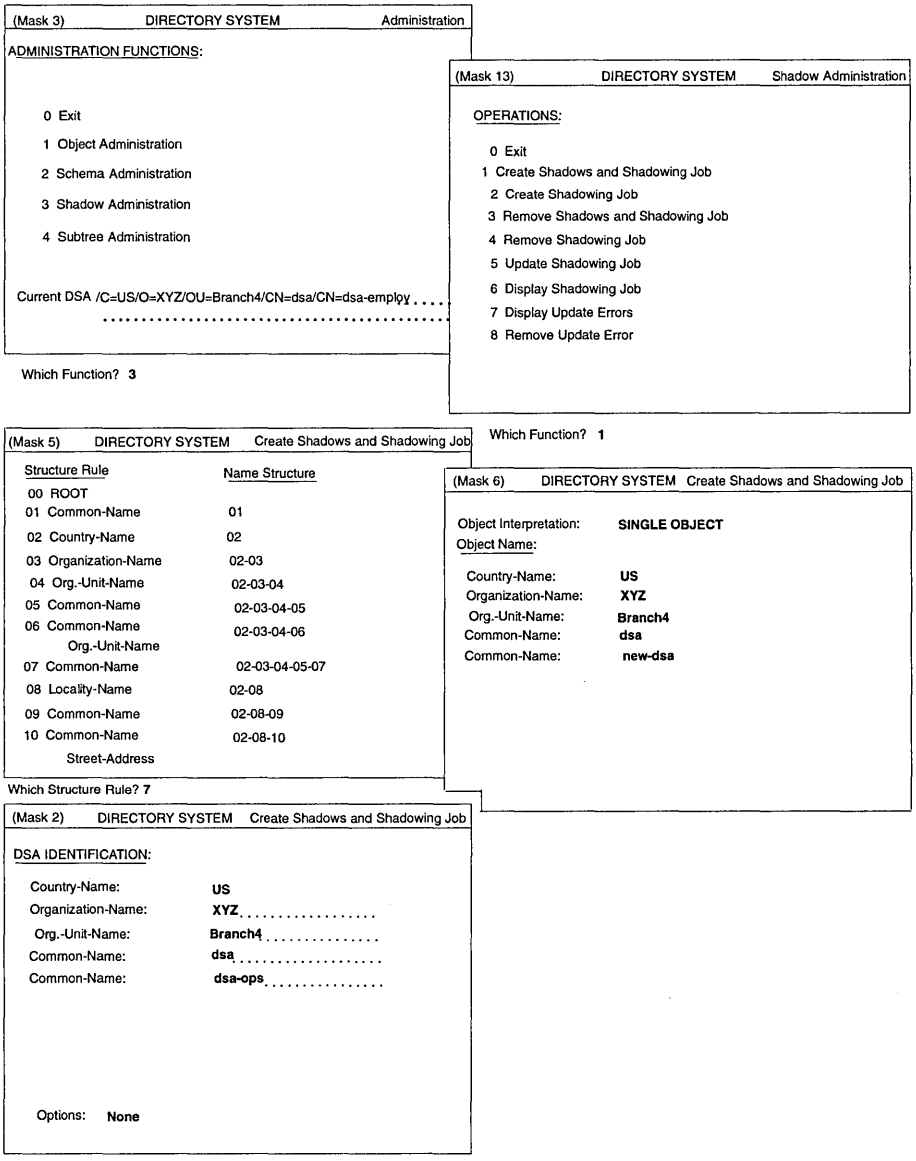
Mask 14e Enter the frequency in days per week.

Mask 14f Enter the frequency in a day of the week (and in an hour of that day).

You are then returned to Mask 2, in which a further target DSA name can be specified. If no more DSAs have to be specified, press **<Del>**.

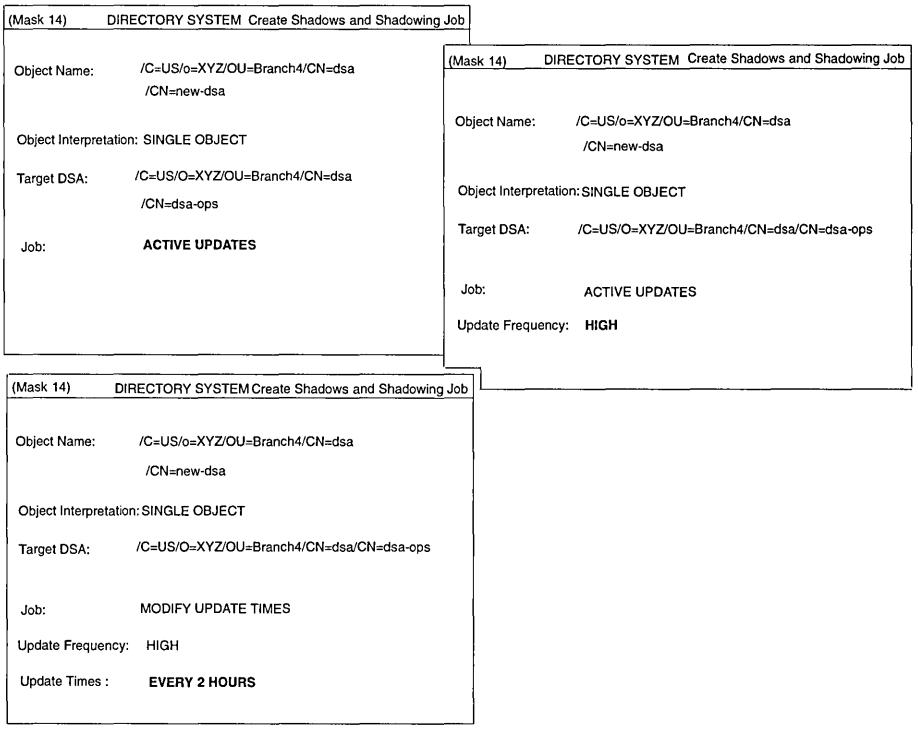
Figure 10-18 shows a sample mask sequence that creates a shadow for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa**. The DN of the current DSA is **C=US/0=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. The shadow is created on the target DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**. The shadowing job is already set to an active state with **MEDIUM** update frequency (**EVERY 2 HOURS**).

Figure 10–18. Sample Create Shadows and Shadowing Job Operation (Part 1)



After Mask 2 in Figure 10-18, Mask 14 is displayed as shown in Figure 10-19. The administrator selects **ACTIVATE UPDATES**. The **Update Frequency** prompt appears. The administrator toggles the space bar to select **MEDIUM**. The **Update Times** prompt appears. The adminstrator selects **EVERY 2 HOURS**. After Mask 14, Mask 2 is displayed to allow the administrator to define the next target DSA. The administrator presses **<Del>** to terminate the operation.

Figure 10–19. Sample Create Shadows and Shadowing Job Operation (Part 2)



10.6.3 Create Shadowing Job

The Create Shadowing Job operation generates an active or inactive shadowing job. In contrast with the Create Shadows and Shadowing Jobs operation, no shadow entries are created. Only use this operation if all the shadows of the subtree already exist in the target DSA (for example, all the directory data has been stored on diskette, and this diskette has been restored on the target DSA).

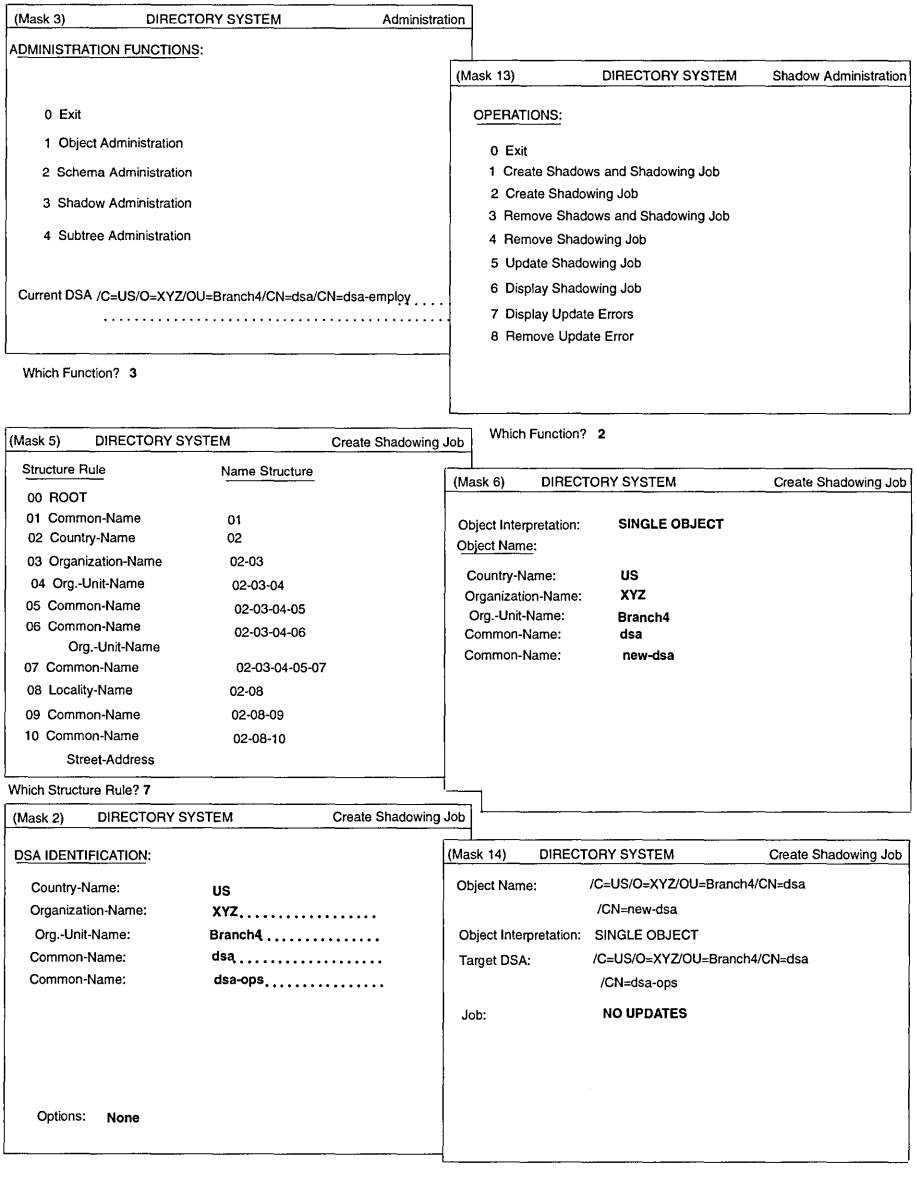
Mask sequence

Mask 3	Select option number 3 (see Chapter 7).
Mask 13	Select option number 2.
Mask 5	Select the structure rule of the object or the root of the subtree whose master entries are to be copied.
Mask 6	Enter the object name of the object or the root of the subtree whose master entries are to be copied.
Mask 2	Enter the DN of the target DSA.
Mask 14a	Select the job administration function from the **Job** field. If you select **ACTIVATE UPDATES** from the **Job** field, Mask 14b is displayed.
Mask 14b	Select the update frequency. Depending on the format of the update times, Mask 14c, 14d, or 14e (followed by Mask 14f) is displayed.
Mask 14c	Enter the frequency in minutes.
Mask 14d	Enter the frequency in hours.
Mask 14e	Enter the frequency in days per week.
Mask 14f	Enter the frequency in a day of the week (and in an hour of that day).

You are then returned to Mask 2, in which a further target DSA name can be specified. If no more DSAs have to be specified, press **<Del>**.

Figure 10-20 shows a sample mask sequence that creates an inactive shadowing job for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa**. The DN of the current DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. The target DSA has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 10–20. Sample Create Shadowing Job Operation



10.6.4 Remove Shadows and Shadowing Job

The Remove Shadows and Shadowing Job operation removes the shadowing job and the shadow entries in the specified target DSA.

Mask Sequence 1

Use this mask sequence if you want to use the Display Shadowing Jobs function to select the shadowing job to be removed.

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 6.

Mask 14 Depending on the format of the update frequency of the shadowing job, the shadowing job to be removed is displayed in Mask 14g, 14h, 14i, or 14j.

Select the shadowing job to be removed by marking it (see Section 10.6.7).

Mask 13 Select option number 3.

Mask Sequence 2

This mask sequence is used to directly specify the shadowing job to be removed.

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 3.

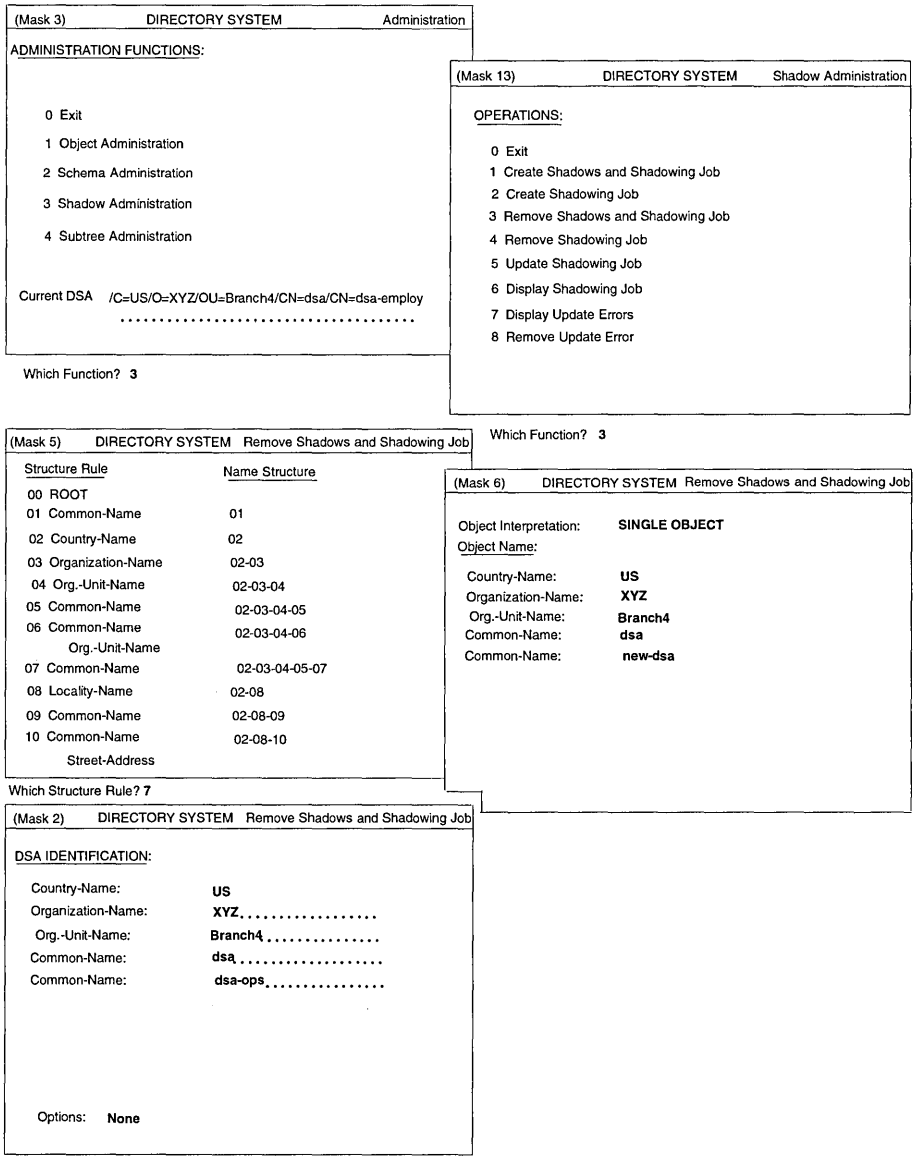
Mask 5 Select the structure rule of the object or the root of the subtree whose shadowing job is to be removed.

Mask 6 Enter the object name of the object or the root of the subtree whose shadowing job is to be removed.

Mask 2 Enter the DN of the target DSA of the shadowing job to be removed.

Figure 10-21 shows a sample mask sequence for Mask Sequence 2, which directly specifies a shadowing job to be removed. The shadowing job for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is removed. The DN of the current DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. The shadow is removed from the target DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 10–21. Sample Remove Shadows and Shadowing Job Operation



10.6.5 Remove Shadowing Job

The Remove Shadowing Job operation removes a shadowing job. In contrast with the previous operation, no shadow entries are removed in the target DSA.

Mask Sequence 1

Use this mask sequence if you want to use the Display Shadowing Jobs function to select the shadowing job to be removed.

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 3.

Mask 14 Depending on the format of the update frequency of the shadowing job, the shadowing job to be removed is displayed in Mask 14g, 14h, 14i, or 14j.

Select the shadowing job to be removed by marking it (see Section 10.6.7).

Mask 13 Select option number 4.

Mask Sequence 2

Use this mask sequence to directly specify the shadowing job to be removed.

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 4.

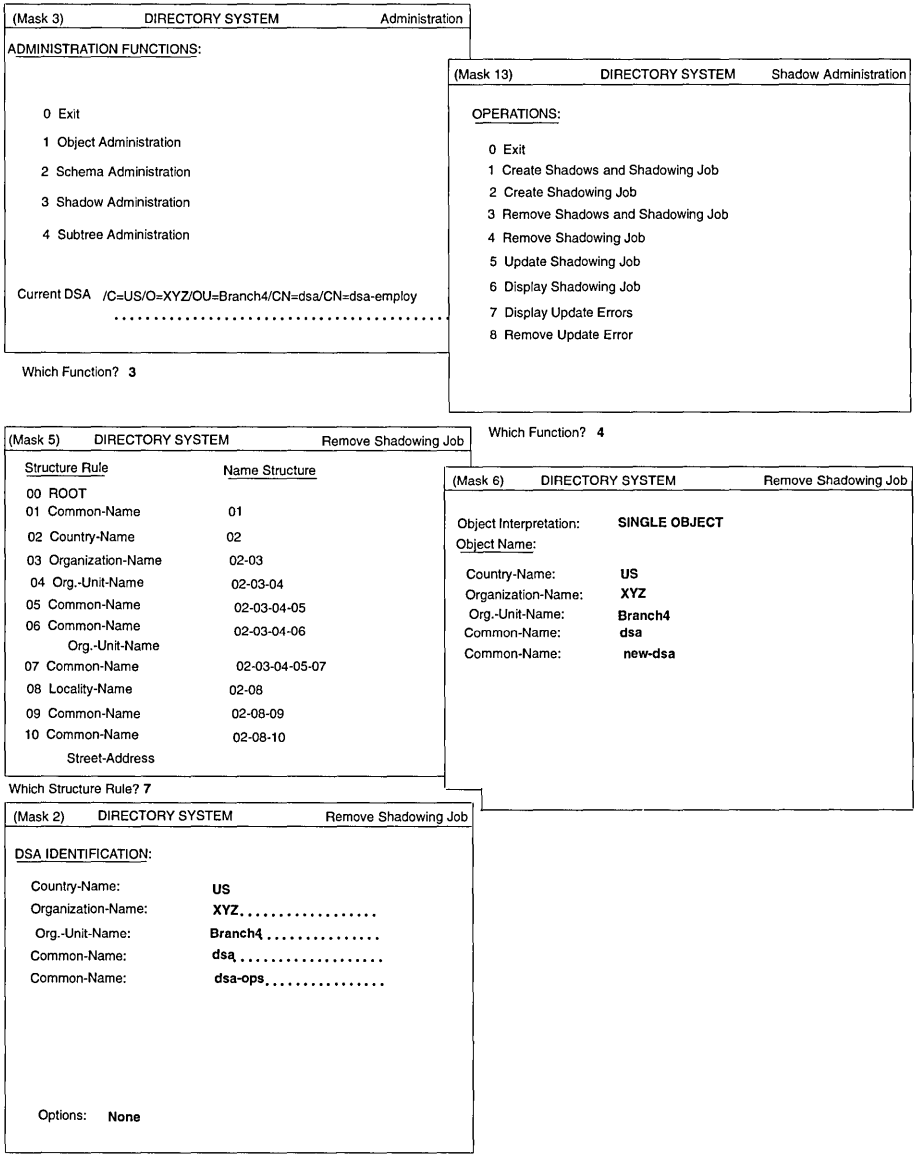
Mask 5 Select the structure rule of the object or the root of the subtree whose shadowing job is to be removed.

Mask 6 Enter the object name of the object or the root of the subtree whose shadowing job is to be removed.

Mask 2 Enter the DN of the target DSA of the shadowing job to be removed.

Figure 10-22 shows a sample mask sequence for Mask Sequence 2, which directly specifies a shadowing job to be removed. The shadowing job for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is removed. The DN of the current DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. The shadow is removed from the target DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 10–22. Sample Remove Shadowing Job Operation



10.6.6 Update Shadowing Job

The Update Shadowing Job operation activates or deactivates a shadowing job or changes the update frequency of an active shadowing job.

Mask Sequence 1

Use this mask sequence if you wish to use the Display Shadowing Jobs function to select the shadowing job to be changed.

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 6.

Select the shadowing job to be changed by marking it (see Section 10.6.7).

Depending on the format of the update frequency of the shadowing job, the shadowing job selected is displayed in Mask 14g, 14h, 14i, or 14j.

Mask 13 Select option number 5.

Mask 14a Select the required job administration function from the **Jobs** field.

If you select **ACTIVATE UPDATES IMMEDIATELY**, the updates are propagated to the target DSA.

If you select **ACTIVATE UPDATES** or **MODIFY UPDATE FREQUENCY**, then Mask 14b is displayed.

Mask 14b Select the update frequency.

Depending on the format of the update frequency, Mask 14c, 14d, or 14e (followed by Mask 14f) is displayed.

Mask 14c Enter the frequency in minutes.

Mask 14d Enter the frequency in hours.

Mask 14e Enter the frequency in days per week.

Mask 14f Enter the frequency in a day of the week (and in an hour of the day).

Mask Sequence 2

Use this mask sequence to directly specify the shadowing job to be changed.

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 5.

Mask 5 Select the structure rule of the object or the root of the subtree whose shadowing job is to be changed.

Mask 6 Enter the object name of the object or the root of the subtree whose shadowing job is to be changed.

Mask 2 Enter the DN of the target DSA of the shadowing job to be changed.

Mask 14a Select the required job administration function from the **Jobs** field.

If you select **ACTIVATE UPDATES IMMEDIATELY**, the updates are propagated to the target DSA.

If you select **ACTIVATE UPDATES** or **MODIFY UPDATE FREQUENCY**, then Mask 14b is displayed.

Mask 14b Select the update frequency.

Depending on the format of the update times, Mask 14c, 14d, or 14e (followed by Mask 14f) is displayed.

Mask 14c Enter the frequency in minutes.

Mask 14d Enter the frequency in hours.

Mask 14e Enter the frequency in days per week.

Mask 14f Enter the frequency in a day of the week (and in an hour of that day).

Figures 10-23 and 10-24 show a sample mask sequence for Mask Sequence 2 that updates a shadowing job. The shadowing job for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** and for target DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is updated. The DN of the current DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. The new update frequency is every five minutes.

Figure 10–23. Sample Update Shadowing Job Operation (Part 1)

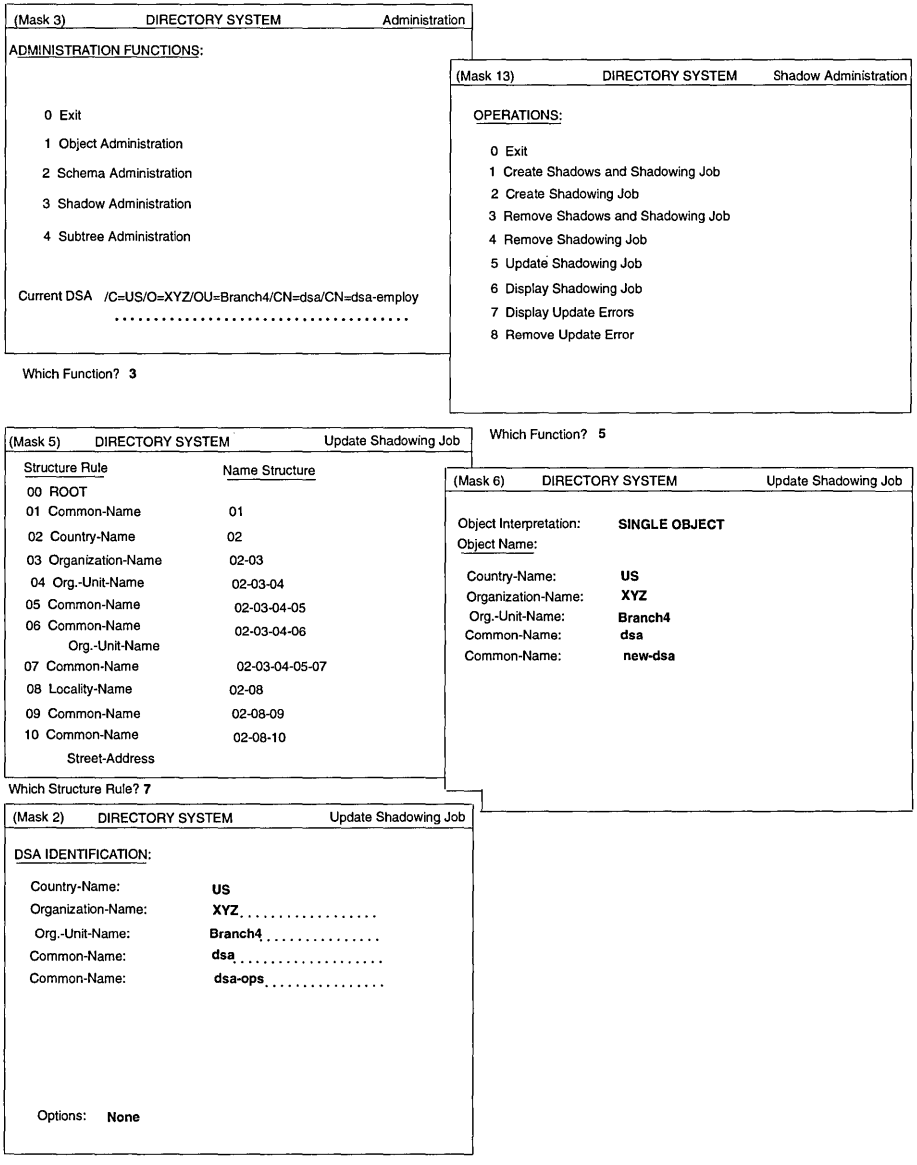
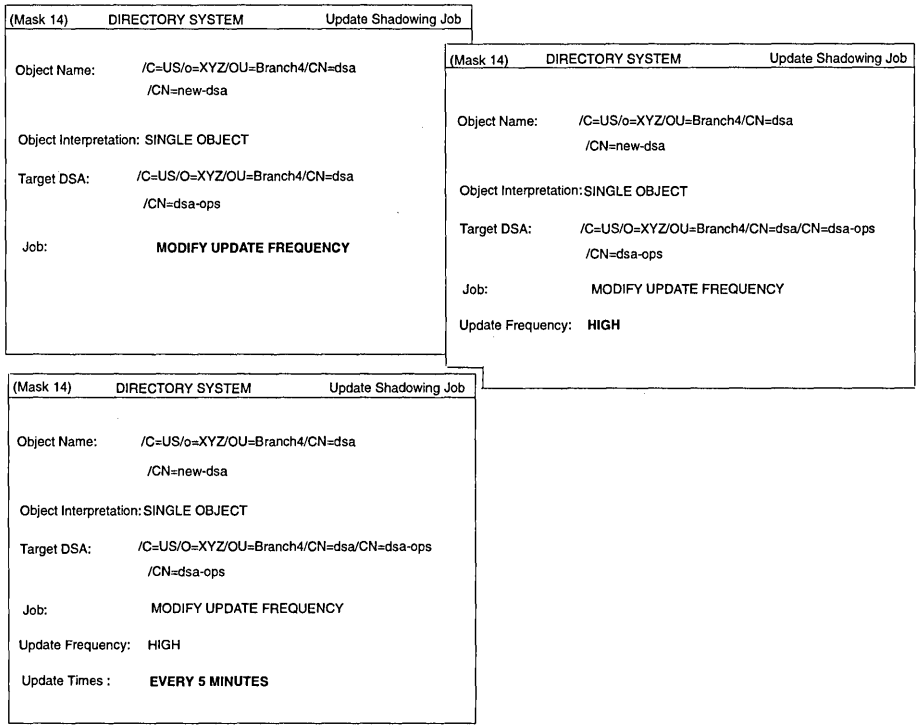


Figure 10–24. Sample Update Shadowing Job Operation (Part 2)



10.6.7 Display Shadowing Jobs

The Display Shadowing Jobs operation displays the existing shadowing jobs.

Mask sequence

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 6.

Depending on the update frequency, Mask 14g, 14h, 14i, or 14j displays the shadowing job.

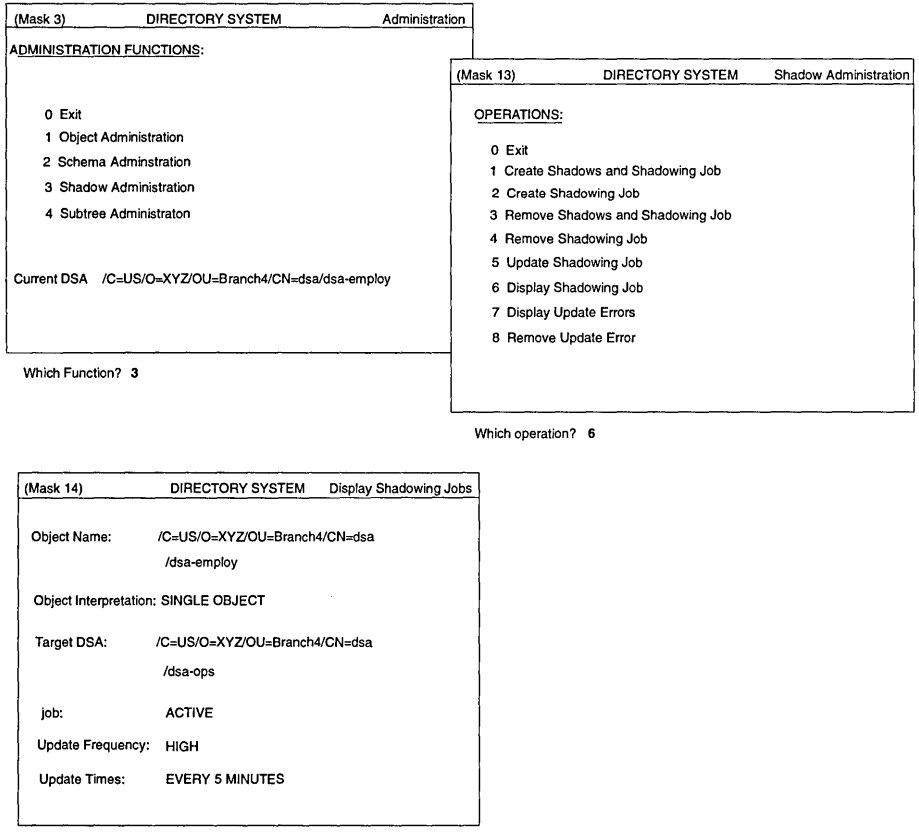
To display the next shadowing job, press **<Menu>** or **<Scroll Down>**.

To mark the shadowing job currently displayed, press **<Return>**. The operation is terminated and the shadowing job marked can be used as input for the operations Remove Shadows and Shadowing Job, Remove Shadowing Job, or Update Shadowing Job.

If no job is to be marked, terminate the operation by pressing **<Del>**.

Figure 10-25 shows a sample mask sequence that displays an active shadowing job with update frequency set to **HIGH (EVERY 5 MINUTES)**. The shadowing job is set for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** for the target DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**. The DN of the current DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**.

Figure 10–25. Sample Display Shadowing Jobs Operation



10.6.8 Display Update Errors

The Display Update Errors operation displays the update errors.

Mask sequence

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 7.

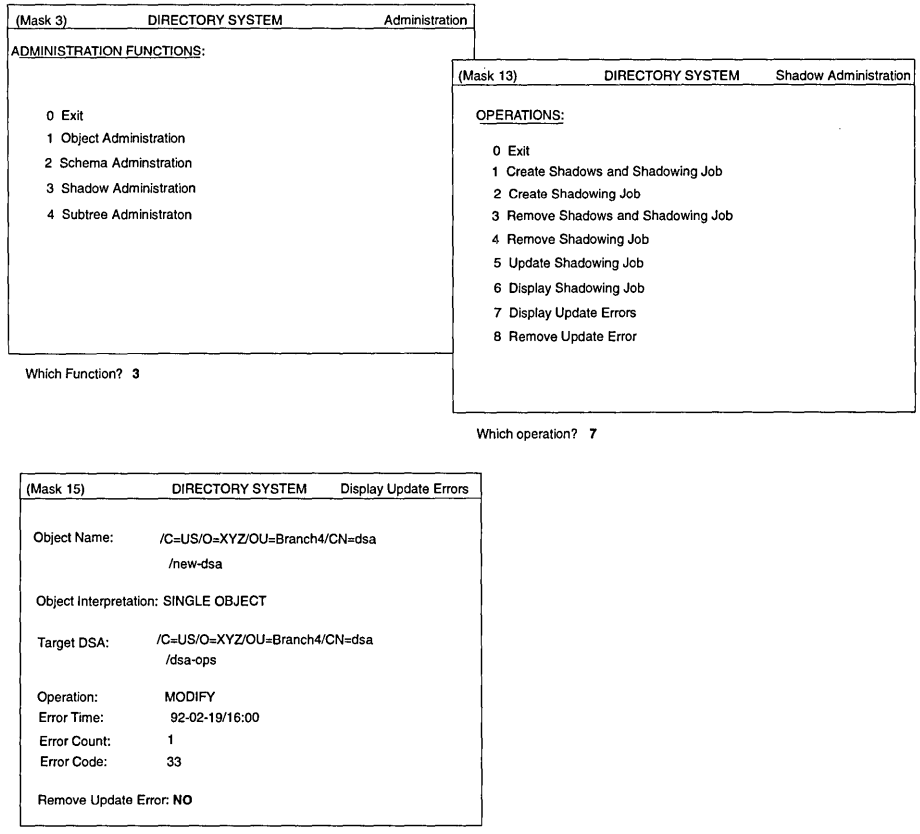
Mask 15 Displays the update error.

To display the next update error, press **<Menu>**, **<Scroll Down>**, or **<Return>**.

Terminate the operation by pressing **<Del>**.

Figure 10-26 shows a sample mask sequence that displays update errors of the shadowing job for the object with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa**. The target DSA has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 10–26. Sample Display Update Errors Operation



10.6.9 Remove Update Errors

The Remove Update Errors operation removes an update error.

Mask sequence

Mask 3 Select option number 3 (see Chapter 7).

Mask 13 Select option number 8.

Mask 15 Displays the update error.

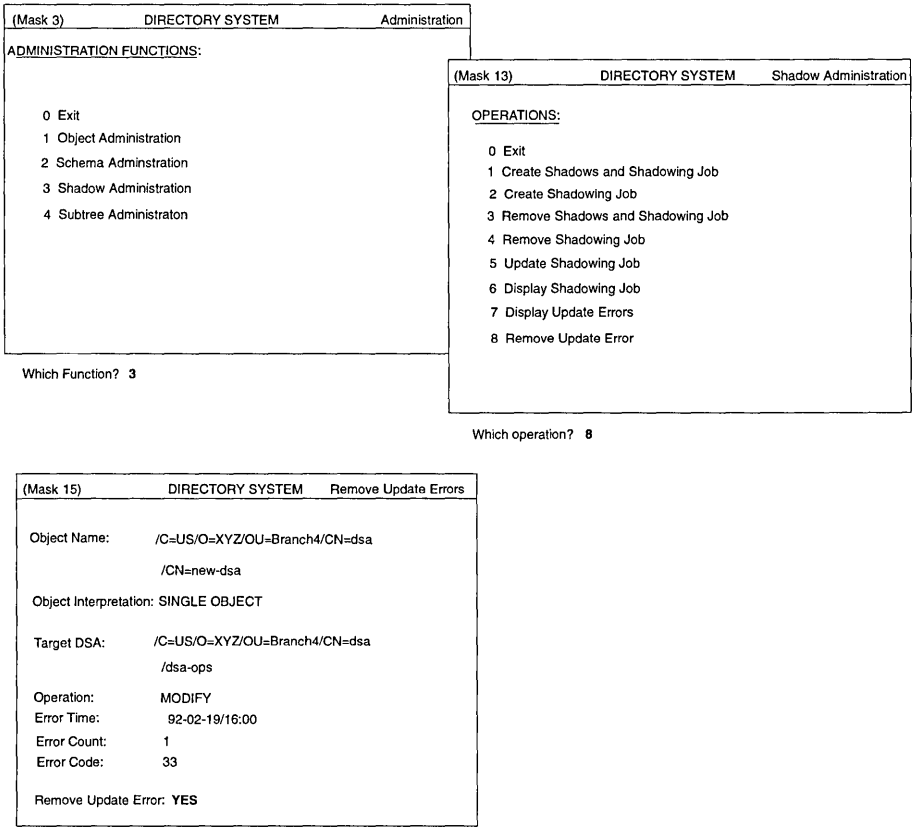
If **Remove Update Error** is set to **YES**, this update error is ignored by the delta update process at the next activation time (otherwise, the delta update process tries to resend the erroneous update operation to the target DSA).

To display the next update error, press **<Menu>**, **<Scroll Down>**, or **<Return>**.

Terminate the operation by pressing **<Del>**.

Figure 10-27 shows a sample mask sequence that removes an update error for a shadowing job. The update error is an attribute error, **33=Attribute missing**, which could occur if the object on the target DSA no longer contains the specified attribute because it has been removed.

Figure 10–27. Sample Remove Update Errors Operation



Chapter 11

Subtree Administration

The subtree administration functions allow the administrator to operate on a subtree of the Directory Information Tree (DIT). An administrator can use these functions to manipulate a large number of objects belonging to a subtree using a single interface. This chapter describes the interface and operation of these functions.

An administrator can perform the following subtree administration operations:

- Save a subtree (object information taken from master DSAs or from a specific DSA) to a file (which could later be used in an Append Subtree operation)
- Append a saved subtree under the same node or under a new node on the same or a new DSA
- Copy a subtree from one node to the same node or to a new node on the same or a new DSA (Save Subtree and Append Subtree operations)
- Change the name of a nonleaf node of the DIT so that name of the object and of all the objects in the subtree are changed
- Delete the subtree from the DIT

- Change the master DSA for objects in a subtree (valid only for those objects of a subtree that have master entries on the specified former DSA)
- Change the attribute value for objects in a subtree that have the specified old value for the attribute (Modify Subtree operation)

If an error occurs during execution of a subtree administration function, the operation may not be completed and the DIT could be left in an inconsistent state. Administrators should make it a practice, when a subtree administration operation fails, to check the state of the DIT.

Subtree administration functions have been implemented using object administration functions, which makes their successful completion subject to access control. An administrator's credentials are established from the logon mask (Mask 1). In general, the administrator must have that following:

- **READ** access rights for all the attributes when an operation involves reading an object (for example, the Save Subtree operation)
- **MODIFY** access rights for the naming attribute(s) of a parent object when adding an object (for example, the Append Subtree operation)
- **MODIFY** access rights for any attribute that is to be modified (for example, in Modify Subtree and Append Subtree operations)

In summary, an administrator should have **READ** access rights for all the attributes of all the objects of the subtree that involve the Save Subtree operation and **MODIFY** access rights for all other functions.

If an administrator is adding objects (implicitly) that exist in the target subtree using the Append Subtree, Copy Subtree, or Change Name/Move Subtree operations, and the attributes of the target objects are different from the attributes of the source objects, the list of such objects is displayed (in Mask 20) either as overwritten or nonoverwritten objects. In Mask 17b (Additional Parameters (Part 2)), an administrator specifies if entries are to be overwritten.

11.1 Change Name/Move Subtree, Append Subtree, and Copy Subtree

The basic difference between Change Name/Move Subtree and Append Subtree and Copy Subtree is that Append Subtree and Copy Subtree operations retain the source subtree. Change Name/Move Subtree deletes the source tree.

Figure 11-1 illustrates the result of a Change Name/Move Subtree operation where **/C=US/O=Sales** is changed to **/C=de/O=Sales**.

Figure 11–1. Moving a Subtree

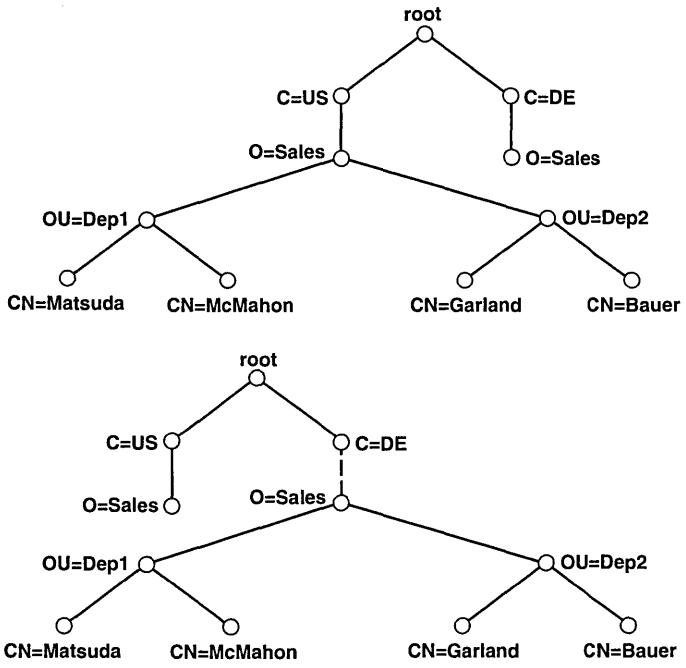
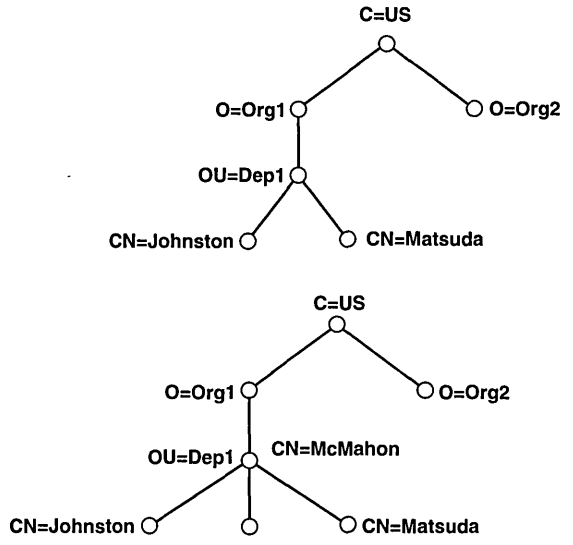


Figure 11-2 illustrates how a sequence of Save Subtree and Append Subtree works when a subtree is saved and appended under the same parent node.

Figure 11–2. Appending a Subtree Under the Same Parent Node

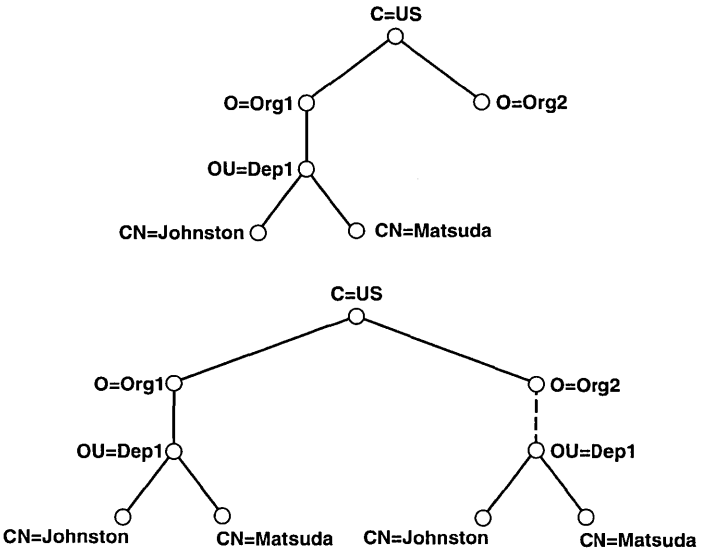


In Figure 11-2,

- The subtree whose root is **/C=US/O=Org1/OU=Dep1** is saved to a file. (**/C=US/O=Org1/OU=Dep1** must be specified as root of the subtree in Mask 6.)
- The entry **/C=US/O=Org1/OU=Dep1/CN=McMahon** is then added.
- The subtree is later appended under the same parent node. (**/C=US/O=Org1** must be specified as the root of the subtree in Mask 6.)
- The object with DN **/C=US/O=Org1/OU=Dep1/CN=McMahon** keeps its **Master Knowledge** attribute, whereas the objects with DN **/C=US/O=Org1/OU=Dep1/CN=Johnston** and DN/**C=US/O=Org1/OU=Dep1/CN=Matsuda** take their **Master Knowledge** attributes from the save file.

Figure 11-3 illustrates how a sequence of Save Subtree and Append Subtree works when the subtree is appended under a node other than the one under which it is saved.

Figure 11–3. Appending a Subtree Under a New Parent Node



In Figure 11-3,

- The subtree whose root is **/C=US/O=Org1/OU=Dep1** is saved to a file.
- The subtree is later appended under a new parent node **/C=US/O=Org2**.

All the entries in the subtree have their **Master Knowledge** attributes set to the DSA where the subtree is appended.

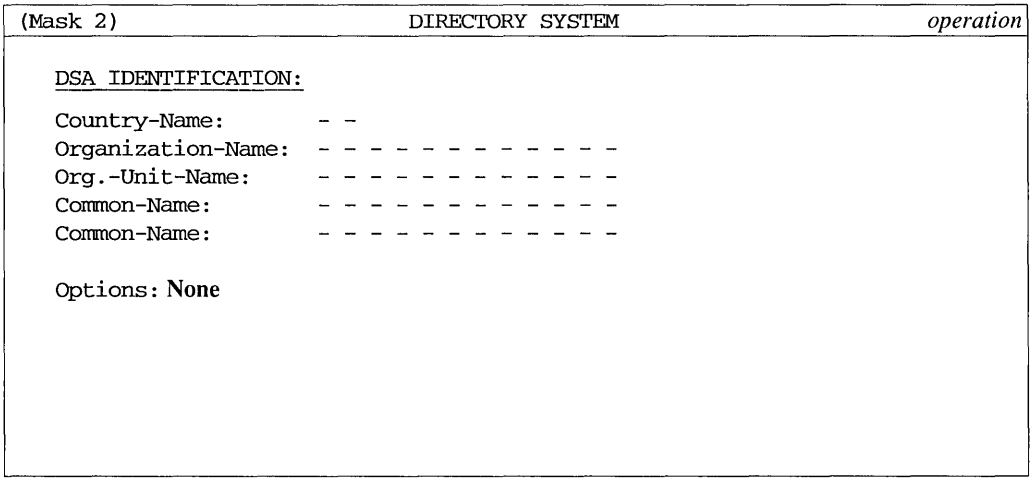
11.2 Masks

Each of the following sections deals with one of the masks used for subtree administration.

11.2.1 Mask 2: DSA Identification

Before a specific source and target DSA for the **Subtree Administration** functions can be specified, their DNs must be entered in Mask 2 (Figure 11-4).

Figure 11–4. Mask 2: DSA Identification



In Mask 2, *operation* will be one of the following values depending on the operation being performed:

Save Subtree
Append Subtree
Copy Subtree
Delete Subtree

Mask 2 displays the following fields:

Country-Name
Enter the **Country-Name** part of the DSA's DN.

Organization-Name
Enter the **Organization-Name** part of the DSA's DN.

Org.-Unit-Name
Enter the **Org.-Unit-Name** part of the DSA's DN.

Common-Name
Enter the **Common-Name** part of the DSA's DN.

Common-Name
Enter the **Common-Name** part of the DSA's DN.

Options Select one of the following options by pressing the space bar:

- **None** (default)
- **Changing Name Structure**

If **Changing Name Structure** is selected, all structure rules are displayed in Mask 5. Only structure rules that represent a DSA can be selected.

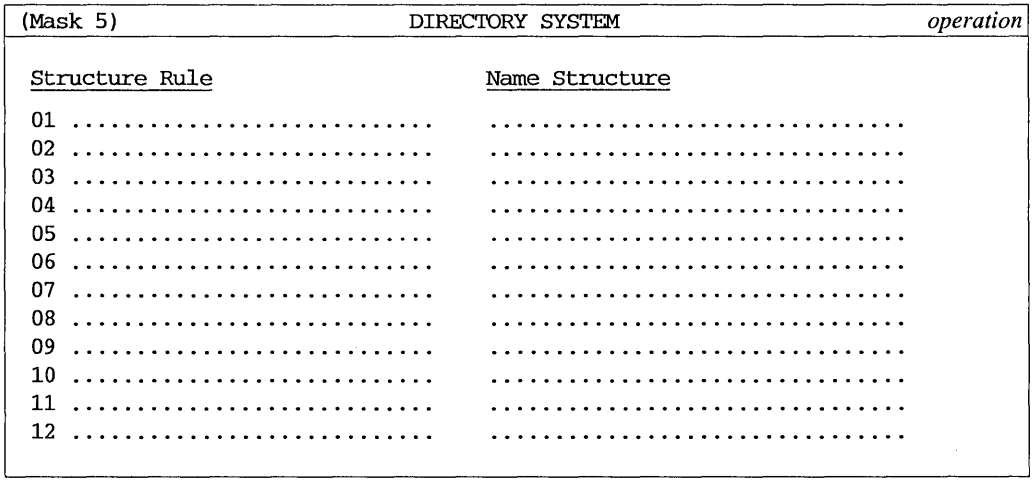
The only structure in the default schema that represents a DSA is structure rule 7.

After the selection is made, Mask 2 displays the selected name structure for entering the DN. This is the same as in shadow administration.

11.2.2 Mask 5: Structure Rule

Mask 5 (Figure 11-5) is used to select the structure rule to be processed. The names of all structure rules stored in the SRT and their name structures are displayed here. If the SRT contains more than 12 structure rules, **<Scroll Up>** and **<Scroll Down>** can be used to page through the mask.

Figure 11–5. Mask 5: Structure Rule



In Mask 5, *operation* will be one of the following values depending on the operation being performed:

Save Subtree
Append Subtree
Copy Subtree
Change Name/Move Subtree
Delete Subtree
Change Master
Modify Subtree

Mask 5 displays the following field:

Which Structure Rule ?
Enter the option number of the structure rule to be processed.

Figure 11-6 shows how the structure rules are displayed in Mask 5.

Figure 11–6. Mask 5: Sample Structure Rules

```
(Mask 5)                    DIRECTORY SYSTEM                    operation

Structure Rule                          Name Structure

01 Common-Name                          01
02 County-Name                          02
03 Organization-Name                    02-03
04 Org.-Unit-Name                       02-03-04
05 Common Name                          02-03-04-05
06 Common Name                          02-03-04-06
     Org.-Unit-Name
07 Common Name                          02-03-04-05-07
08 Locality-Name                        02-08
09 Common-Name                          02-08-09
10 Common-Name                          02-08-10
     Street-Address

Which Structure Rule ?
```

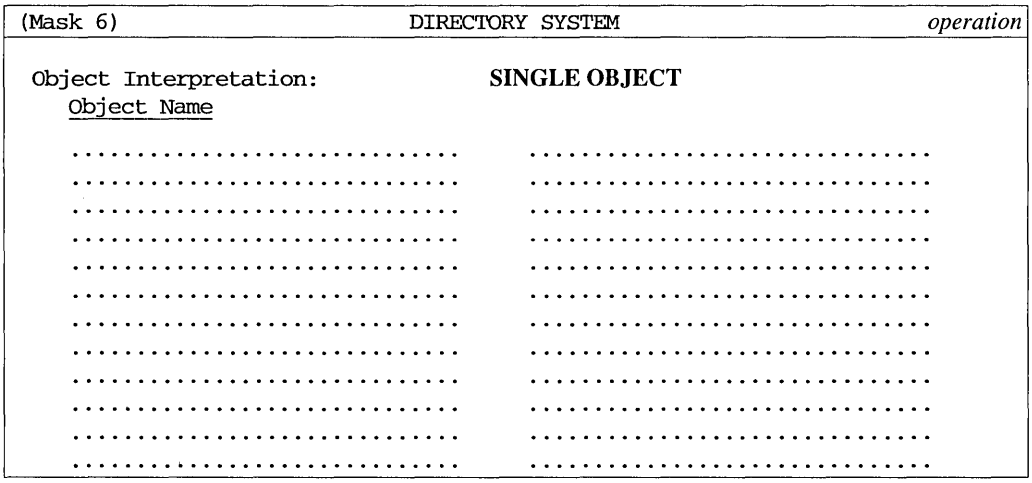
00 Root is not displayed if the structure rule for the DN of **Target DSA** or **Source DSA** must be entered.

11.2.3 Mask 6: Object Name

Mask 6 (Figure 11-7) is used to input the object name or subtree name for the subtree administration.

Wildcards are not permitted when defining subtrees.

Figure 11–7. Mask 6: Object Name



In Mask 6, *operation* will be one of the following values depending on the operation being performed:

Save Subtree
Append Subtree
Copy Subtree
Change Name/Move Subtree
Delete Subtree
Change Master
Modify Subtree

Mask 6 displays the following fields:

Object Interpretation

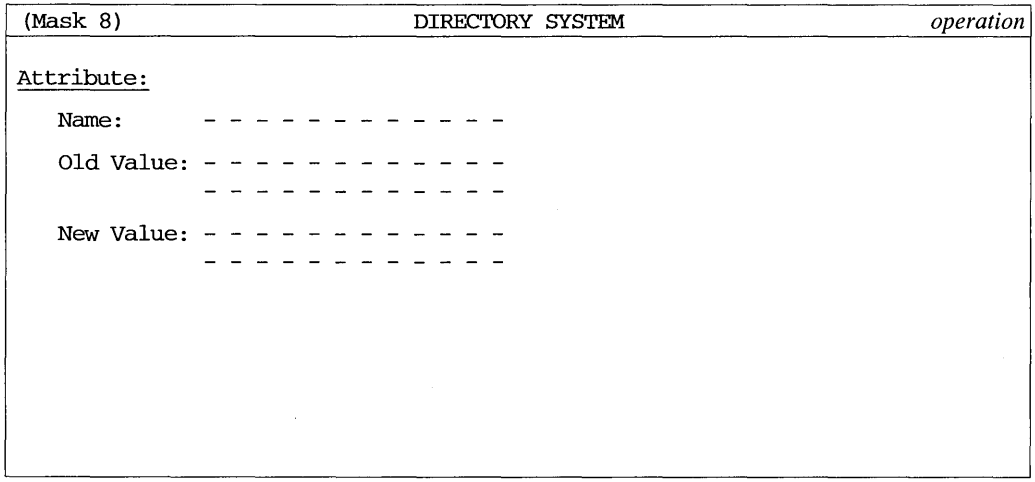
Select one of the following object interpretations by pressing the space bar:

- **SINGLE OBJECT**
- **OBJECT AND ITS SUBORDINATES**

11.2.4 Mask 8: Attribute (Modify)

Mask 8 (Figure 11-8) is used to change either the **Master Knowledge** attribute if *operation* is **Change Master**, or any other attribute if *operation* is **Modify Subtree**, (except **Presentation-Address**, **Master Knowledge**, or attributes with Preferred Delivery Method syntax, ASN1 syntax, Any syntax, or Search Guide syntax).

Figure 11–8. Mask 8: Attribute (Modify)



In Mask 8, *operation* will be one of the following depending on the operation being performed:

Change Master
Modify Subtree

If *operation* is **Change Master**, the name **Master Knowledge** is displayed in the **Name** field, and the old master DSA is displayed both in the **Old Value** and **New Value** fields. Only the **New Value** field must be overwritten by the new master DSA.

Mask 8 displays the following fields:

Name Enter the name of the attribute to be modified. This value is only valid if *operation* is **Modify Subtree**.

Old Value Enter the old value of the attribute. This value is only valid if *operation* is **Modify Subtree**.)

New Value If *operation* is **Change Master**, enter the new master DSA, followed by an ' (apostrophe) to enable a blank to be entered as the last value character.

If *operation* is **Modify Subtree**, enter the new attribute value.

If the syntax is Boolean, the value can be **TRUE** or **FALSE**.

If the syntax is Preferred Delivery Method, the integers are entered separated by a space.

If the syntax is Distinguished Name, the name must be entered with no leading spaces. Separate the DNs with commas. Do not use a space before or after a comma.

For example:

/C=de/O=Smith Ltd/OU=dep1/CN=Huber,OU=AP11

The existence of an object with the DN entered is not checked.

The following attributes have their own masks (see Chapter 8).

- **CDS-Cell** (Mask 21)
- **CDS-Replica** (Mask 22) (where CDS refers to Cell Directory Service)

The attributes with the following syntax also have their own masks:

- Postal Address syntax (Mask 25)
- MHS O/R Address syntax (Mask 27)
- MHS O/R Address syntax (Mnemonic) (Mask 28)
- MHS O/R Address syntax (Numeric) (Mask 29)
- MHS O/R Address syntax (Structured Postal) (Mask 30)
- MHS O/R Address syntax (Unstructured Postal) (Mask 31)
- MHS O/R Address syntax (Terminal) (Mask 32)
- MHS DL Submit Permission syntax (Mask 33)
- MHS DL Submit Permission syntax (Mask 34) (Individual, Member of DL, Pattern Match)

- MHS DL Submit Permission syntax (Member of Group) (Mask 35)
- MHS O/R Name syntax (Mask 34)

Only the name needs to be entered for these attributes. (The input for the values of these attributes is ignored in Mask 8.)

11.2.5 Mask 16: Subtree Operations

Mask 16 (Figure 11-9) is used to specify the subtree administration operation to be executed.

Figure 11–9. Mask 16: Subtree Operations

```
(Mask 16)                    DIRECTORY SYSTEM          Subtree Administration

    OPERATIONS

      0 Exit
      1 Save Subtree
      2 Append Subtree
      3 Copy Subtree
      4 Change Name/Move Subtree
      5 Delete Subtree
      6 Change Master
      7 Modify Subtree
```

```
Which operation ?
```

Mask 16 displays the following field:

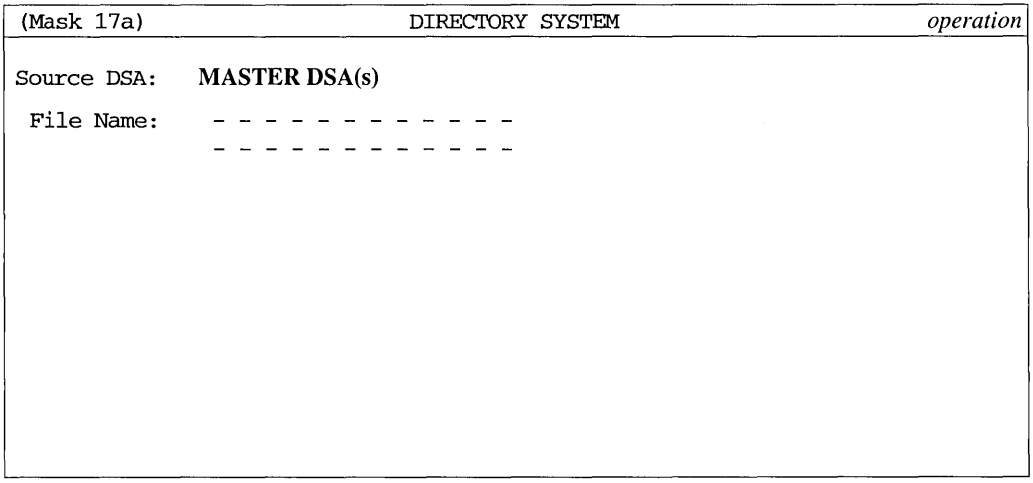
Which operation ?

Enter the number of the operation to be executed (0 to 7).

11.2.6 Mask 17a: Additional Parameters (Part 1)

Mask 17a (Figure 11-10) is used to input additional parameters required by the subtree administration operations.

Figure 11–10. Mask 17a: Additional Parameters (Part 1)



In Mask 17a, *operation* will be one of the following values depending on the operation being performed:

Save Subtree
Copy Subtree
Delete Subtree

Mask 17a displays the following fields:

Source DSA

Select one of the following values by pressing the space bar:

- **MASTER DSA(s)** (default)
- **BIND DSA**
- **SPECIFIC DSA**

File Name

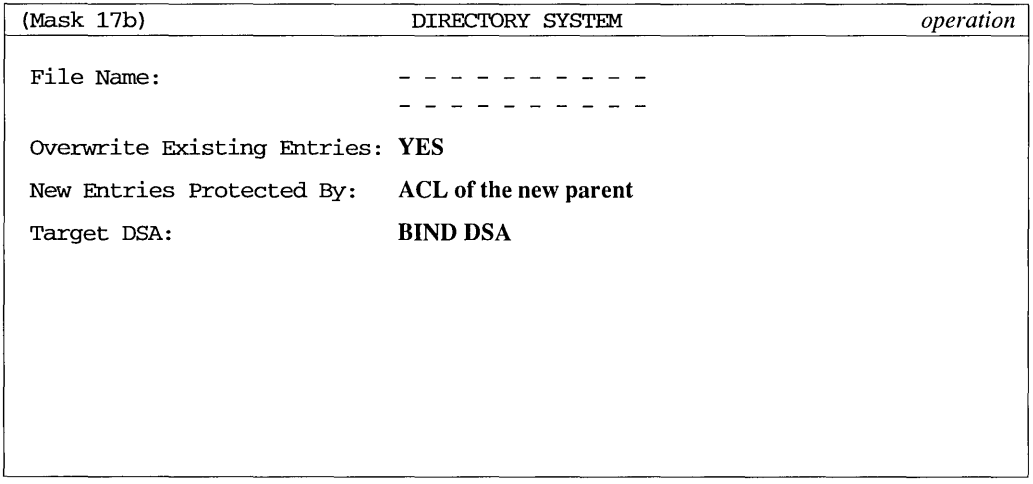
Enter the name of the file in which the objects are to be saved.

If *operation* is **Copy Subtree** or **Delete Subtree**, the **File Name** field is not displayed in the mask.

11.2.7 Mask 17b: Additional Parameters (Part 2)

Mask 17b (Figure 11-11) is used to input additional parameters required by the subtree administration operations.

Figure 11–11. Mask 17b: Additional Parameters (Part 2)



In Mask 17b, *operation* will be one of the following values depending on the operation being performed:

Append Subtree
Copy Subtree

Mask 17b displays the following fields:

File Name Enter the filename of file in which the objects to be appended were stored by **Save Subtree**.

Overwrite Existing Entries
Select one of the following values by pressing the space bar:

- **NO**
- **YES** (default)

New Entries Protected By
Select one of the following values by pressing the space bar:

- **ACL of the new parent** (default)
- **Original ACL**

Target DSA Select one of the following values by pressing the space bar:

- **BIND DSA** (default)
- **SPECIFIC DSA**

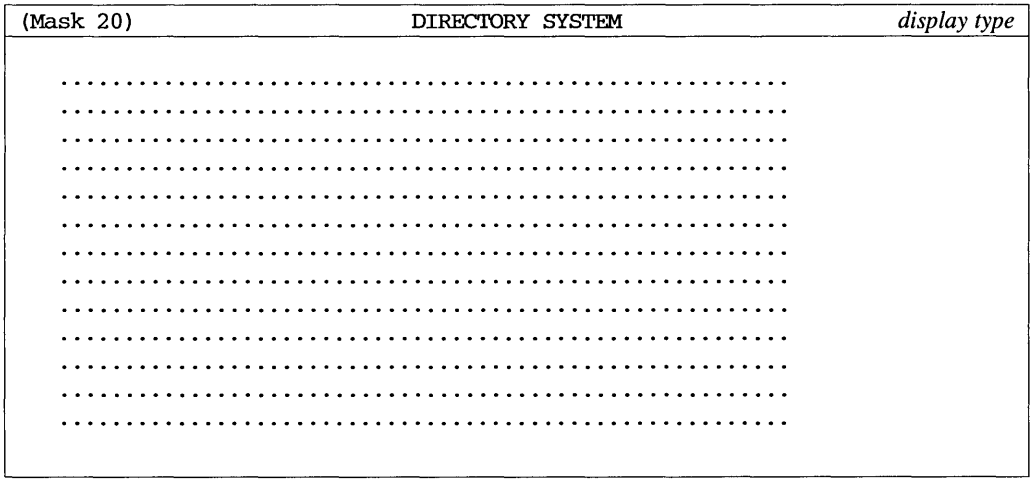
In the case of the Copy Subtree operation, the **File Name** field is not displayed in the mask.

11.2.8 Mask 20: Object List

Mask 20 (Figure 11-12) displays the objects that could not be processed by the called function.

Each element is output in a line, shortened to the mask width if required; for example, **Jones/Dep1/Smith Ltd./US**.

Figure 11–12. Mask 20: Object List



In Mask 20, *display type* will be one of the following values depending on the operation being performed:

Nonoverwritten objects
Overwritten objects
Unchanged objects
Objects not removed

If the list contains more elements than can be held in a mask, press **<Scroll Up>** and **<Scroll Down>** to page up and down.

To exit Mask 20, press **<Del>**.

11.3 Operations

Each of the following sections deals with one of the operations used for subtree administration.

11.3.1 Save Subtree

The Save Subtree operation writes the object entries of a subtree in binary format to a file. The object entry's information could be taken from a master DSA(s), bind DSA, or a specific DSA, depending on the option selected in Mask 17a (Additional Parameters).

In order to save all object information successfully, the administrator must have read access to all attributes of all objects in the subtree. (If the administrator does not have read access, the operation fails to read all objects or all attributes, or both.)

Note: In order to avoid naming conflicts with existing filenames, the name of the save file must be unique. If an existing filename is entered, this file is overwritten.

Mask sequence

Mask 3 Select option number 4.

Mask 16 Select option number 1.

Mask 5 Select the structure rule of the root of the subtree to be saved.

If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the object name of the root of the subtree to be saved.

Mask 17a Select the source DSA from which the objects are to be saved:

MASTER DSA(s)
Save the master entries of the subtree from the master DSA or DSAs.

BIND DSA Save the master or shadow entries of the bind DSA.

SPECIFIC DSA
Save the master or shadow entries of the DSA specified in Mask 2.

Note: The file is created in binary format by this operation. The file is created in the directory from which **gdssysadm** or **gdsditadm** is started, unless a full pathname is given for the file.

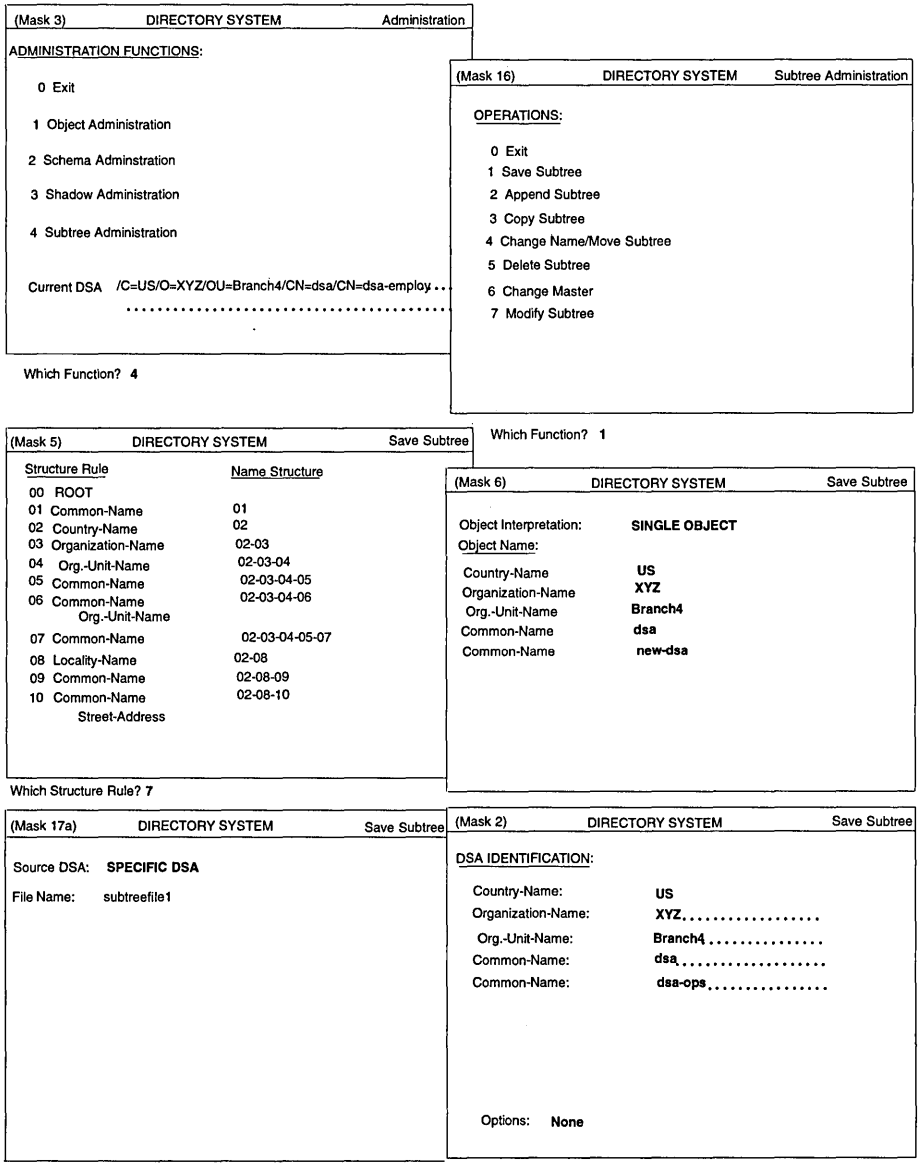
Enter the name of the file in which the objects are to be saved.

If you select **SPECIFIC DSA** as the source, then Mask 2 is displayed.

Mask 2 Enter the DN of the DSA.

Figure 11-13 shows a sample mask sequence that writes the object with the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** to a file named **subtreefile1**. The DN of the source DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**. The current DSA has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. Input is highlighted in bold.

Figure 11–13. Sample Save Subtree Operation



11.3.2 Append Subtree

The Append Subtree operation appends a subtree from a file, which was created beforehand with the Save Subtree function, under a (new) parent node on a target DSA (original or different).

In order to append the objects, the administrator must have modify access to the naming attribute(s) of the root of the subtree. The administrator must also have modify access to all attributes of all objects of the subtree. (If the administrator does not have modify access, the operation fails to append all objects or all attributes, or both.)

In Mask 17b, the administrator can decide whether the appended objects will keep their original ACL or whether they will receive the ACL of the new parent node.

If the subtree is appended under a new parent, then the DSA where the subtree is appended is the master of all entries that are added.

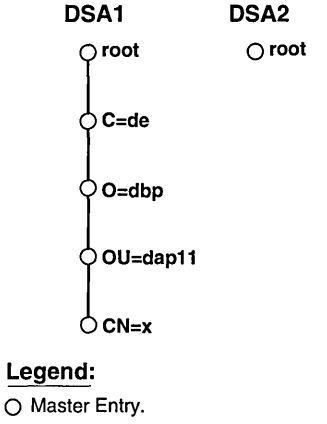
If the subtree is to be appended under the same parent node from where it is saved, objects to be appended that already exist keep the same **Master Knowledge** attribute value. Objects to be appended that do not exist take the value of the **Master Knowledge** attribute from the save file.

An object that already exists under the (new) parent node is never deleted. If such an object does not exist in the saved subtree, the object is not affected by the operation. If the object exists in the saved subtree and has different attribute values, it is or is not overwritten depending on the selection made in Mask 17b.

The new parent node under which the subtree is to be appended must already exist as a master or shadow in the target DSA.

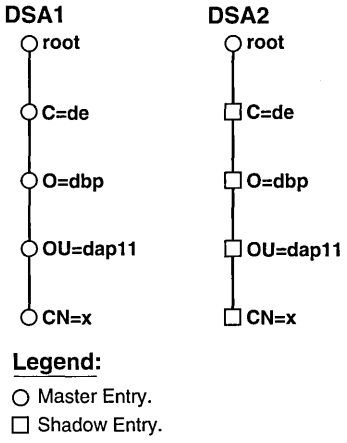
In some cases, the Append Subtree operation leads to the creation or updating of optional shadows. It is the responsibility of the administrator to create shadowing jobs for the shadows that are created by this operation. If some shadows were created by shadow administration operations and were updated by the Append Subtree operation, the shadowing job could fail for these shadows later. These cases are illustrated in the following examples. Figure 11-14 shows two DSAs, **DSA1** and **DSA2**, before the Save Subtree operation.

Figure 11–14. DSA1 and DSA2 Before Save and Append Operations (Example 1)



The /**C=de** subtree is saved in a file using a Save Subtree operation on **DSA1** and later appended under **root** on **DSA2**. The result is shown in Figure 11-15.

Figure 11–15. DSA1 and DSA2 After Save and Append Operations

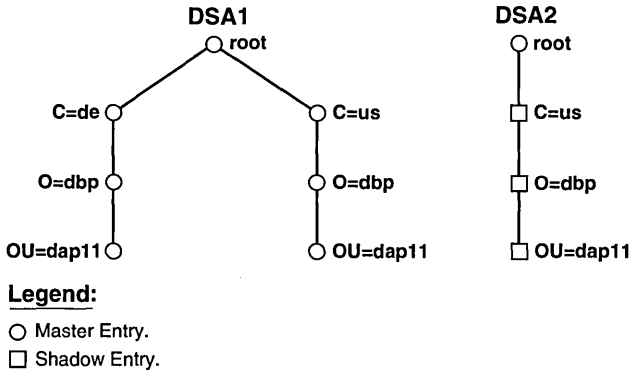


Because shadows have been created on **DSA2**, the administrator should create a shadowing job on **DSA1** for **/C=de** and its subordinates with the target DSA as **DSA2**.

The second example shows how the subtree under **/C=de/O=dbp** is saved on **DSA1** and appended under **/C=us** on **DSA2**. The **C=us** subtree on **DSA2** was created by shadow administration operations and a shadowing job already exists for it.

Figure 11-16 shows **DSA1** and **DSA2** before the Save Subtree and Append Subtree operations.

Figure 11–16. DSA1 and DSA2 Before Save and Append Operations (Example 2)



After the Append Subtree operation, **/C=us/O=dbp** and **/C=us/O=dbp/OU=dap11** are modified by **/C=de/O=dbp** and **/C=de/O=dbp/OU=dap11**, respectively, and not by a shadowing job. The first two objects will be modified with attribute information taken from the last two objects if the attributes of these objects are different. The shadowing job for these shadow entries will fail if the shadows have been modified by this operation. It is recommended that shadows be created and updated by shadow handling operations and not by Append Subtree and Copy Subtree operations.

Mask sequence

Mask 3: Select option number 4.

Mask 16: Select option number 2.

Mask 5: Select the structure rule of the new parent object of the root of the subtree to be added.

If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6: Enter the DN of the new parent object of the root of the subtree to be added.

Mask 17b: Enter the name of the file in which the subtree is saved.

Specify whether existing entries are to be overwritten.

Select the ACL to be used to protect the new entries.

Select the target DSA in which the subtree is to be added.

If you select **SPECIFIC DSA** as the target DSA, then Mask 2 is displayed.

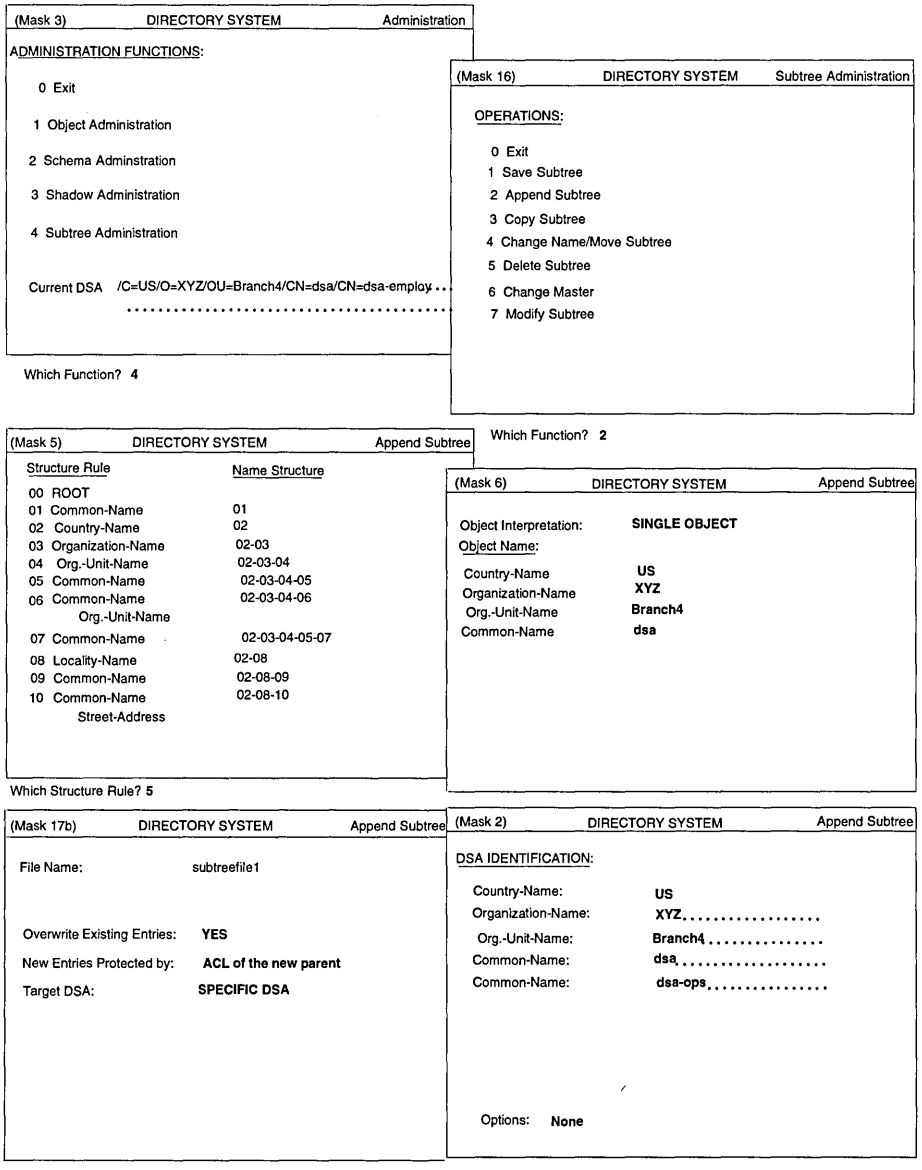
Mask 2: Enter the DN of the DSA.

Note: The file with the subtree to be added must have been created beforehand with the Save Subtree operation. The file is in binary format.

If an object to be added during this operation already exists, but has different attributes, the object is displayed in Mask 20. Depending on the selection in the **Overwrite Existing Entries** field, the *display type* field in Mask 20 is either **Nonoverwritten objects** or **Overwritten objects.**

Figure 11-17 shows a sample mask sequence that appends the object with the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** from the file **subtreefile1**. The DN of the target DSA is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**. The current DSA has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**. Input is highlighted in bold.

Figure 11–17. Sample Append Subtree Operation



After Mask 2, Mask 20 is displayed because the object that was appended (**/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa**) already exists in the DSA **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

11.3.3 Copy Subtree

The Copy Subtree operation copies a subtree from one node to another. The target node (new parent) could be in the same DSA as the source node or exist on a different DSA.

The operation works in the same way as a Save Subtree operation followed by an Append Subtree operation (see Sections 11.3.1 and 11.3.2). The file in which the objects of the subtree are saved during the operation is deleted after the Copy Subtree operation is completed.

Mask sequence

Mask 3 Select option number 4.

Mask 16 Select option number 3.

Mask 5 Select the structure rule of the root of the subtree to be copied. If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the object name of the root of the subtree to be copied.

Mask 17a Select the source DSA as follows:

MASTER DSA(s)
Copy the master information of the subtree.

BIND DSA Copy the information of the BIND DSA.

SPECIFIC DSA
Copy the information of the DSA specified in Mask 2.

If you select **SPECIFIC DSA** as the source, then Mask 2 is displayed.

Mask 2 Enter the DN of the DSA.

Mask 5 Select the structure rule of the parent of the root of the subtree that is to be copied. If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the DN of the parent of the root of the subtree to be copied.

Mask 17b Specify whether existing entries will be overwritten.

Select the ACL to be used to protect the new entries.

Select the target DSA to which the subtree is to be copied.

If you select **SPECIFIC DSA** as the target DSA, then Mask 2 is displayed.

Mask 2 Enter the DN of the DSA.

The **File Name** field is not displayed in Masks 17a and 17b.

If an object to be added during this operation already exists but has different attributes, the object is displayed in Mask 20. Depending on the selection in the **Overwrite Existing Entries** field, the *display type* field in Mask 20 is either **Nonoverwritten objects** or **Overwritten objects**.

Figures 11-18 and 11-19 show a sample mask sequence that copies the object with the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** from the source DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** to the current DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**.

Figure 11–18. Sample Copy Subtree Operation (Part 1)

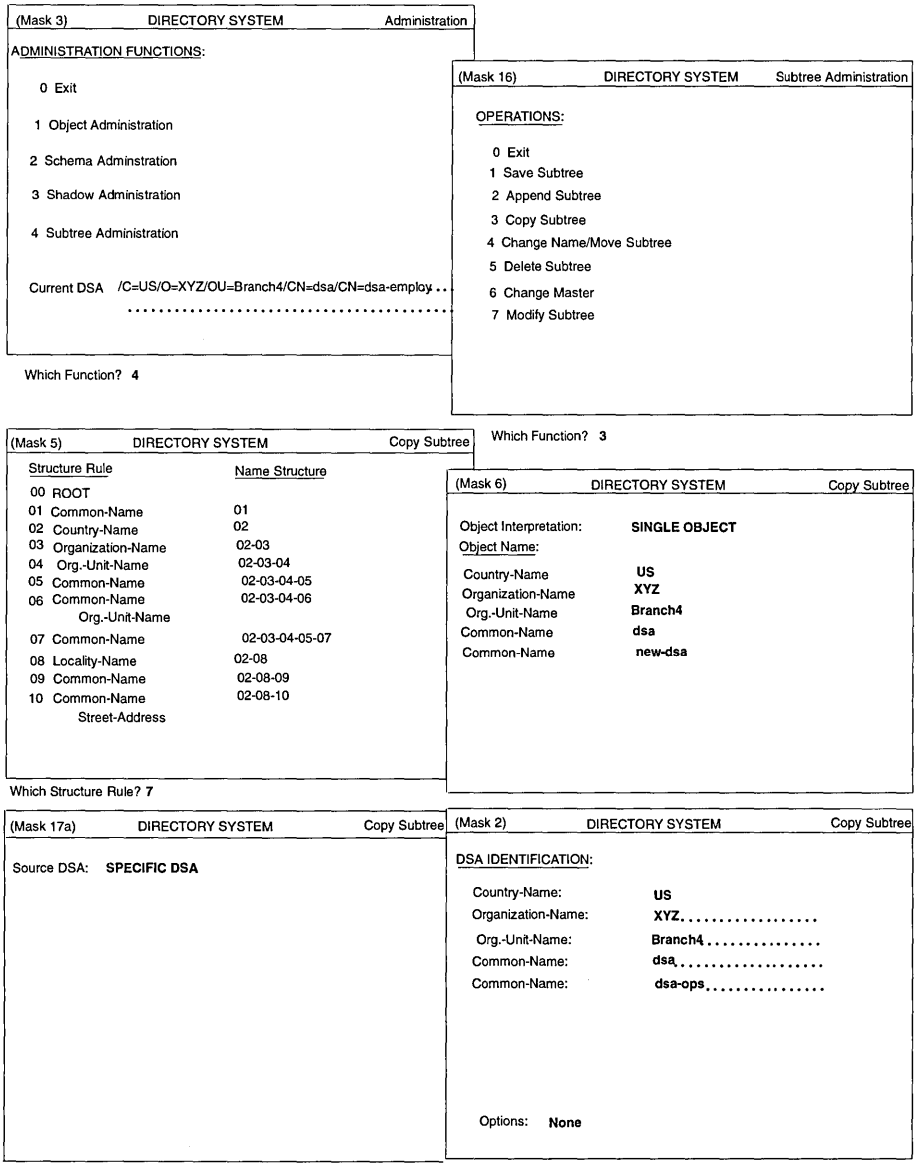
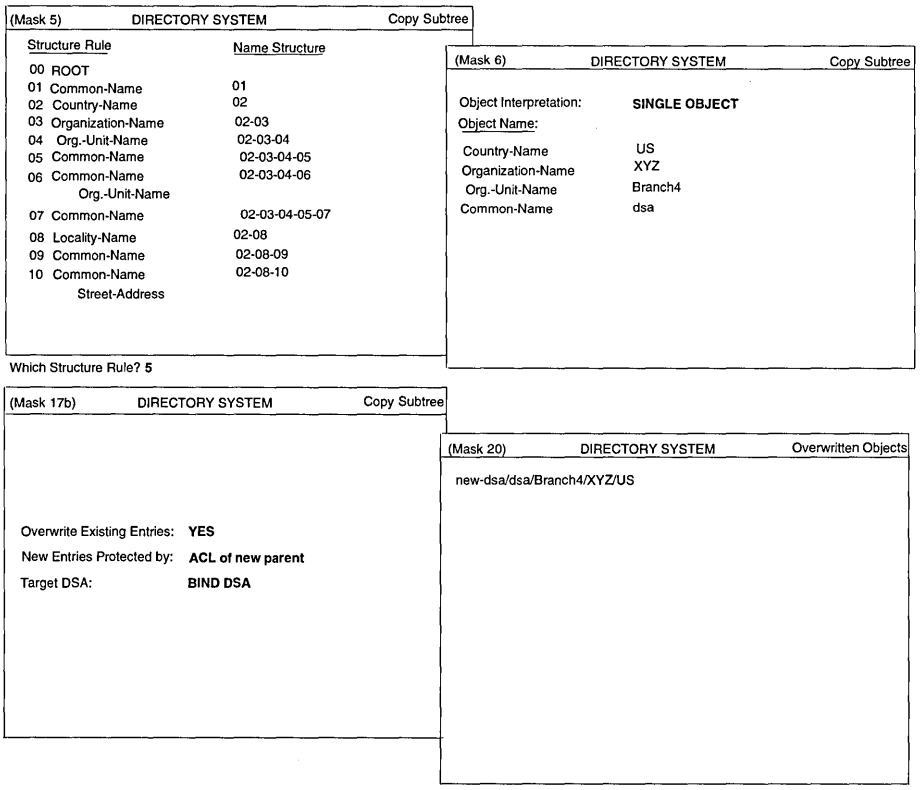


Figure 11–19. Sample Copy Subtree Operation (Part 2)



After Mask 17b, Mask 20 is displayed because the object that has been copied (**/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa**) already exists in the DSA **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**.

11.3.4 Change Name/Move Subtree

The Change Name/Move Subtree operation changes the name of an entry (not necessarily an end node), or moves a subtree. This operation only changes master entries.

If the name of a nonend node is changed, the new name is also valid for its subordinate nodes. The operation does not change any attributes of the objects moved or renamed.

In order to rename or move the objects, the administrator must have modify access to the naming attribute(s) of the parent nodes and to all attributes of all objects of the subtree. (If the administrator does not have modify access, the operation fails to rename or move all objects or all attributes, or both.)

If an object or subtree cannot be renamed or moved completely, the old object or subtree is retained.

The new parent node must already exist as a master or shadow.

Objects that already exist under the new parent node that also exist in the old subtree are overwritten or not overwritten depending on the selection made in Mask 17b and whether these objects also exist in the old subtree. Objects in the new subtree that do not exist in the old subtree are not affected by the operation.

The Change Name/Move Subtree operation typically deletes shadows of the old subtree on certain DSAs and not on others. The shadow update job fails for the shadows that have been deleted. Administrators should run the shadowing jobs once for the shadows that have not been deleted and delete shadowing jobs for the old subtree. Figures 11-20 and 11-21 show the subtrees on three DSAs before and after a Change Name/Move Subtree operation where **/C=us** is changed to **/C=in**.

Figure 11–20. Before a Change Name/Move Subtree Operation

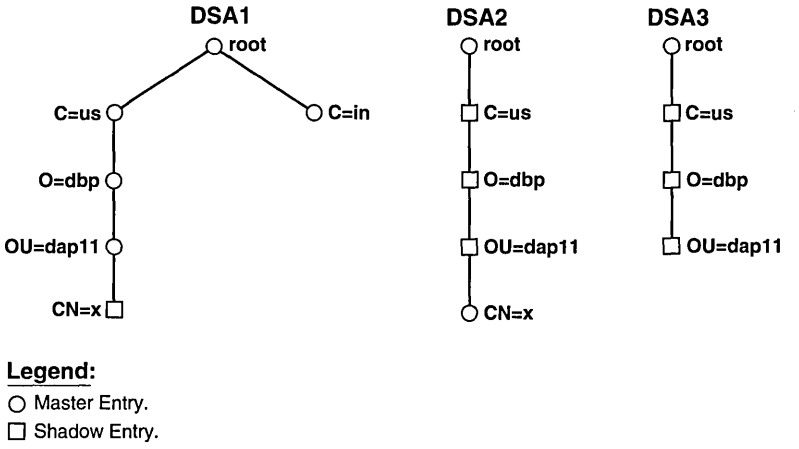
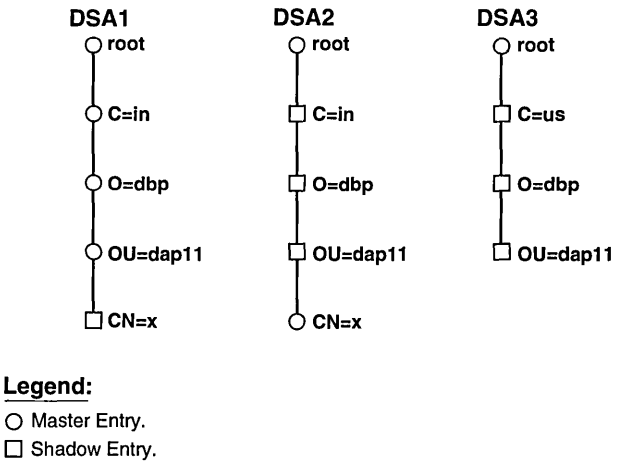


Figure 11–21. After a Change Name/Move Subtree Operation



The shadowing job for **/C=us** (and its subordinates) for **DSA2** will fail. The administrator should run the shadowing job for **DSA3** then delete the shadowing job. The administrator should also create a shadowing job for the **/C=in** subtree for target DSA **DSA2** because shadows have been created by the Change Name/Move Subtree operation. Shadows and a shadowing job for the new subtree should be created for target **DSA3** if the new subtree is to be known on that DSA.

Mask sequence

Mask 3 Select option number 4.

Mask 16 Select option number 4.

Mask 5 Select the old structure rule of the root of the subtree that is to be renamed or moved.

If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the old object name of the root of the subtree that is to be renamed or moved.

Mask 5 Select the structure rule of the (new) subtree root (after renaming or moving). If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the object name of the (new) subtree root (after renaming or moving).

Mask 17b Specify whether existing entries need to be overwritten.

Note: The **File Name**, **Target DSA**, and **New entries protected by** fields are not displayed in Mask 17b.

If some entries cannot be changed or moved, Mask 20 displays a list of unchanged objects.

Figures 11-22 and 11-23 show a sample mask sequence that changes the name of the object entry from **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** to **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-doc**. The current DSA has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**.

Figure 11–22. Sample Change Name/Move Subtree Operation (Part 1)

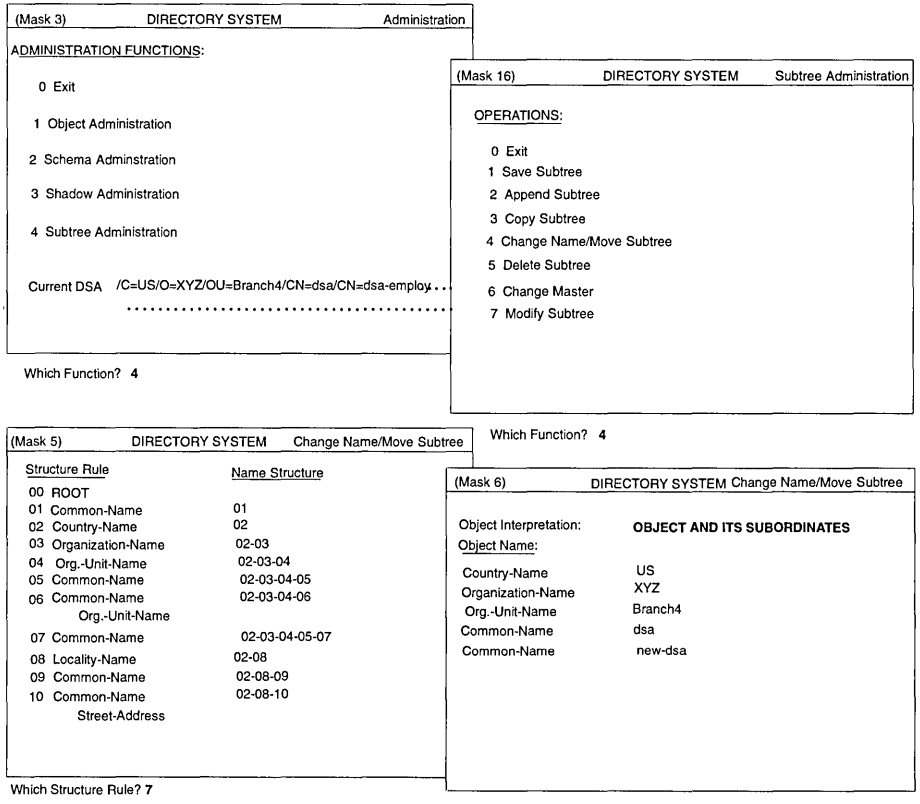
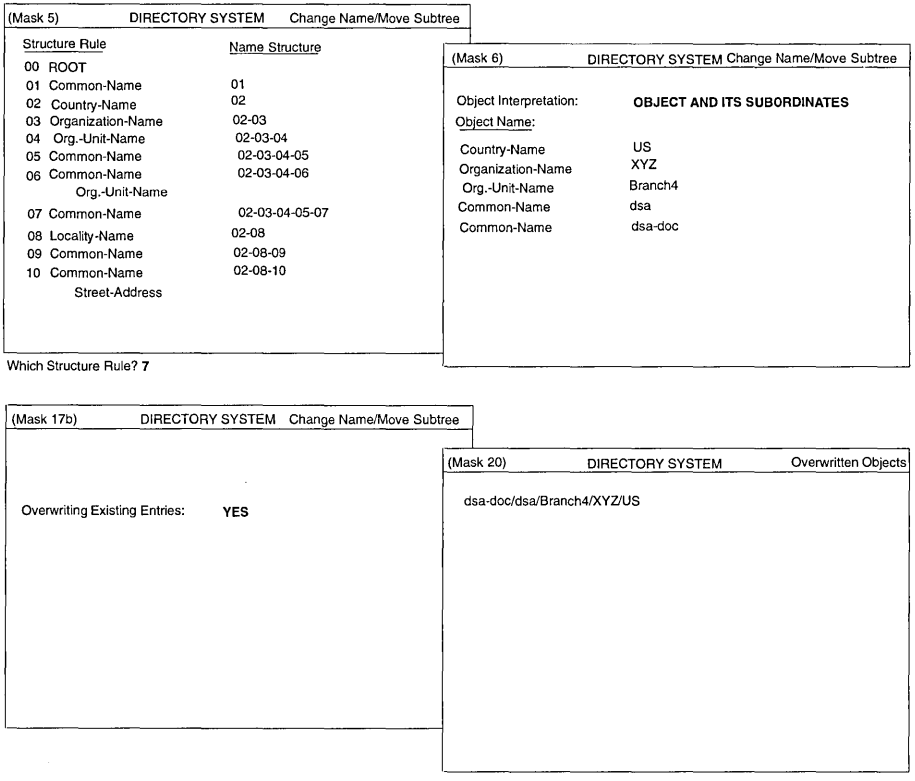


Figure 11–23. Sample Change Name/Move Subtree Operation (Part 2)



After Mask 17b, Mask 20 is displayed because the object that has been moved (**/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa**) already exists with the new name **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-doc**.

11.3.5 Delete Subtree

The Delete Subtree operation deletes a subtree from the master DSA(s), from the bind DSA, or from a specific DSA.

In order to delete the subtree, the administrator must have modify access to the naming attributes of all objects to be deleted.

If master entries have been deleted and shadows and shadowing jobs exist, the shadowing jobs should be run once for all DSAs where shadowing jobs exist. Then these shadowing jobs should be deleted. If shadows have been deleted, the shadowing job for these shadows will fail and should be deleted.

Mask sequence

Mask 3 Select option number 4.

Mask 16 Select option number 5.

Mask 5 Select the structure rule of the root of the subtree to be deleted.

If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the object name of the root of the subtree to be deleted.

Mask 17a Select the source DSA whose objects are to be deleted:

MASTER DSA(s)
Delete the master information of the subtree. (Only master entries are deleted.)

BIND DSA Delete the information of the BIND DSA. (Master and shadow entries are deleted in the BIND DSA.)

SPECIFIC DSA
Delete the information of the DSA specified in Mask 2. (Master and shadow entries are deleted in the specified DSA.)

If you select **SPECIFIC DSA**, then Mask 2 is displayed.

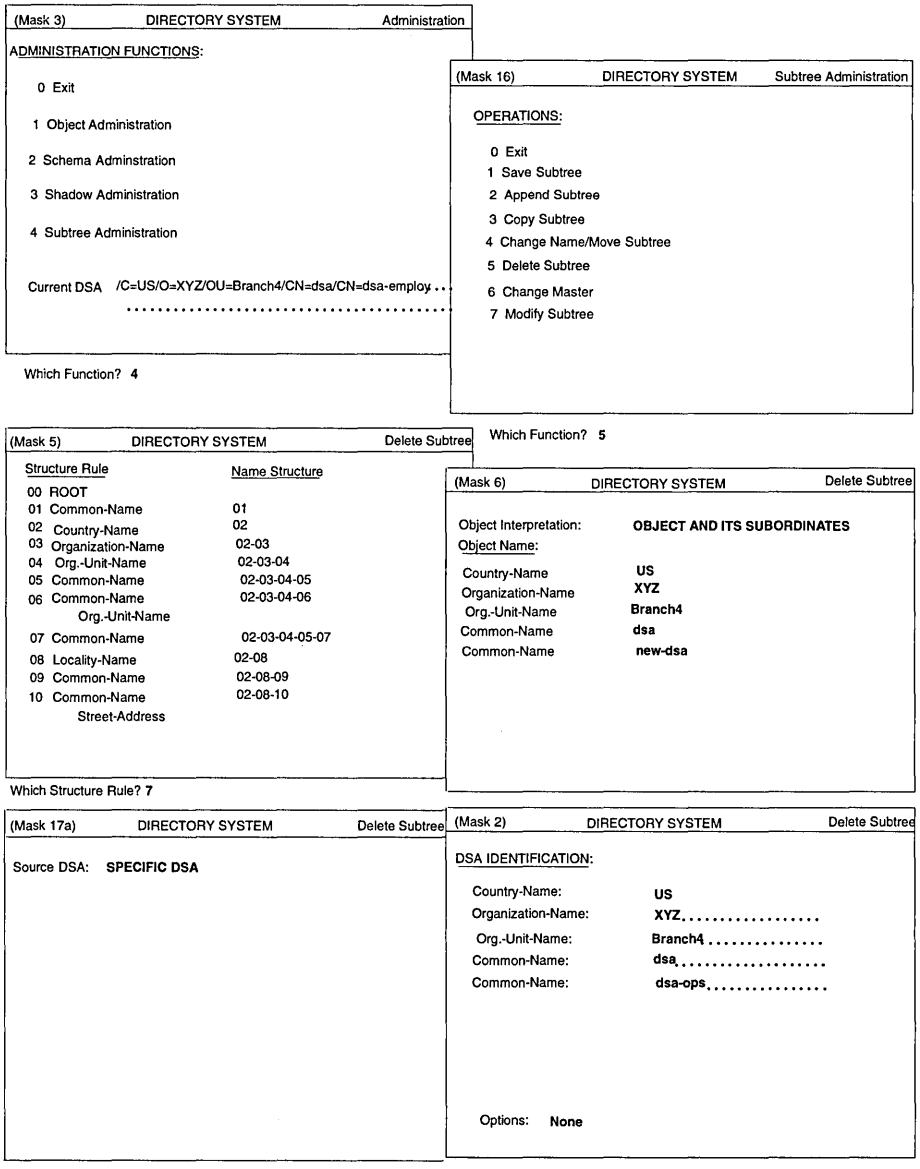
Mask 2 Enter the DN of the DSA.

If the subtree contains objects that cannot be deleted (for example, because of an ACL), then these objects are displayed in Mask 20.

The **File Name** field is not displayed in Mask 17a.

Figure 11-24 shows a sample mask sequence that deletes the name of the object entry with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** from the source DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**. The current DSA has the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ**.

Figure 11–24. Sample Delete Subtree Operation



If the object **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** cannot be deleted, Mask 20 is displayed.

11.3.6 Change Master

The Change Master operation changes the **Master Knowledge** attribute of objects of a subtree. Only those objects that have the specified old value for the **Master Knowledge** attribute are affected by this operation. Therefore, objects that are master entries in the subtree now become shadows of master entries on another DSA (new master DSA).

Before changing the **Master Knowledge** attribute of a subtree, the administrator must ensure that shadow entries exist for all the entries of the subtree in the new master DSA.

The administrator must have modify access to the **Master Knowledge** attribute of all objects where the attribute is to be changed.

Once a **Change Master** operation is complete, the administrator must ensure that shadowing jobs that update the shadows on other DSAs (other than the new master DSA) are run once. Shadowing jobs that update shadows on the new master DSA will fail because those shadows will already have been updated by the **Change Master** operation. Shadowing jobs for old master entries on old master DSAs should be deleted. In addition, shadowing jobs should be created for new master entries on the new master DSA.

Figures 11-25 and 11-26 show three DSAs, **DSA1**, **DSA2**, and **DSA3** before and after a **Change Master** operation.

Figure 11–25. DSA1, DSA2, and DSA3 Before a Change Master Operation

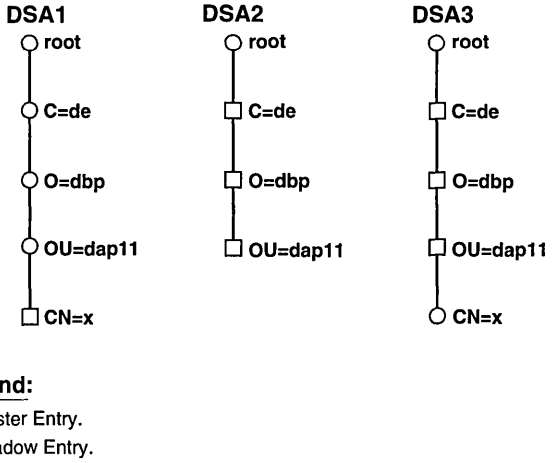
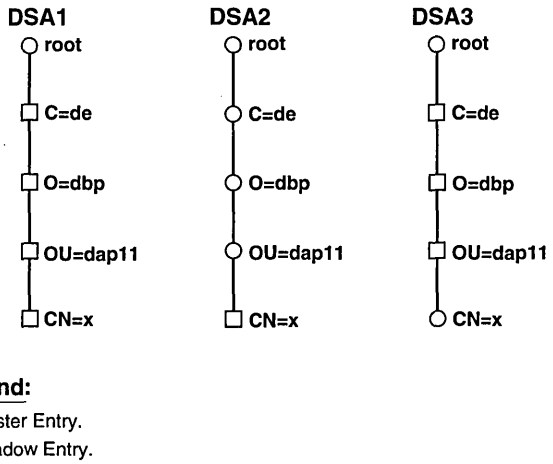


Figure 11–26. DSA1, DSA2, and DSA3 After a Change Master Operation



The shadowing job for **/C=de** (subtree) and **DSA2** (the target DSA for the shadowing job) will fail. The shadowing job for **/C=de** (subtree) and **DSA3** (the target DSA for the shadowing job) must be run. The administrator must delete shadowing jobs for **/C=de** from **DSA1**, must create shadowing jobs for **/C=de** (subtree) on **DSA2** for **DSA1** and **DSA3**, and must create a shadowing job for **/CN=x** (single object) on **DSA3** for **DSA1**.

Mask sequence

Mask 3 Select option number 4.

Mask 16 Select option number 6.

Mask 5 Select the structure rule of the root of the subtree whose **Master Knowledge** attribute is to be changed.

If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.

Mask 6 Enter the object name of the root of the subtree whose objects you want to change the **Master Knowledge** attribute for.

Mask 8 The value **Master Knowledge** is displayed in the **Name** field.

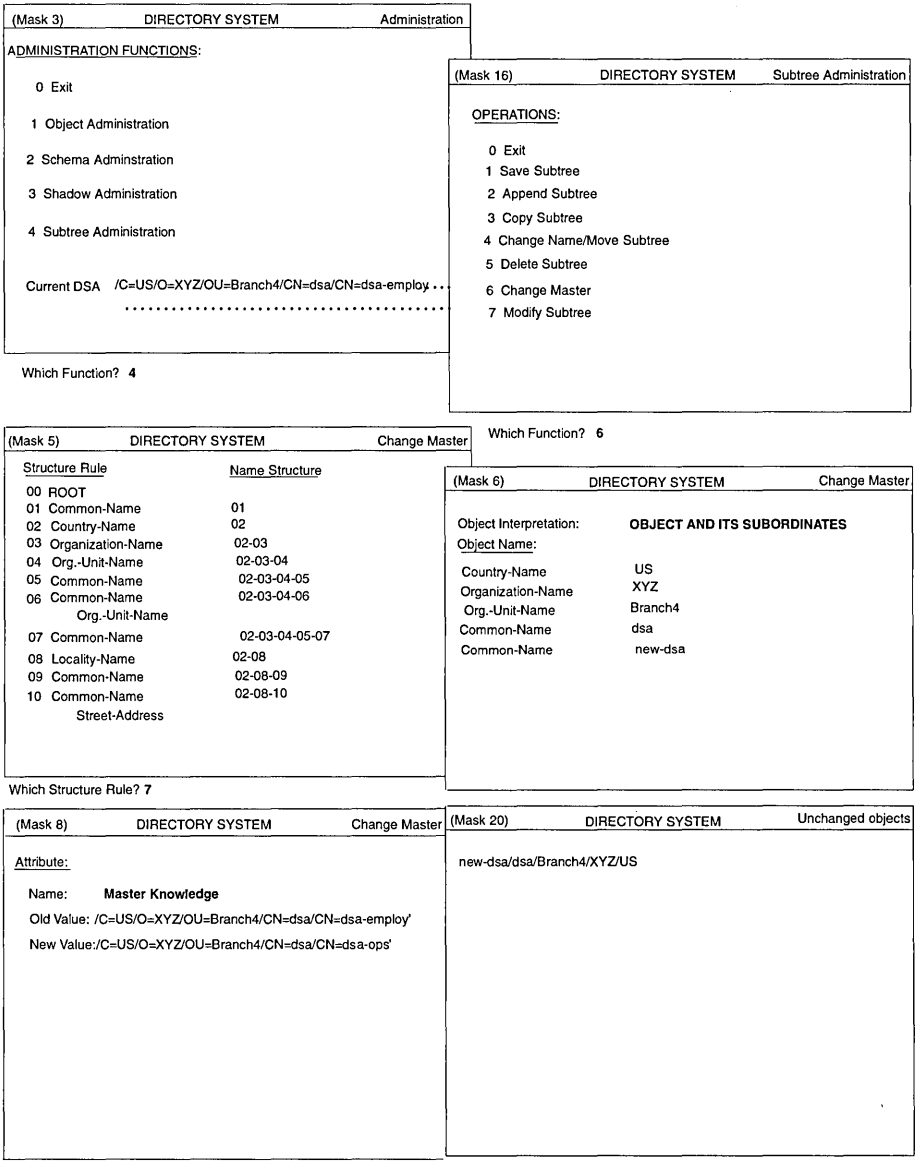
The old value of the DSA is displayed in the **Old Value** and **New Value** fields.

Enter the new master DSA in the **New Value** field.

If any problems arise during this operation (for example, due to incorrect access rights), a list of names that cannot be changed is displayed in Mask 20.

Figure 11-27 shows a sample mask sequence that changes the **Master Knowledge** attribute of the object entry with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa.** The **Master Knowledge** attribute is changed from the DSA with DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** to the DSA with the DN **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops.**

Figure 11–27. Sample Change Master Operation



If the **Master Knowledge** attribute could not be changed (for example, because of ACL restrictions), Mask 20 is displayed with the objects whose **Master Knowledge** attributes could not be changed.

11.3.7 Modify Subtree

The Modify Subtree operation modifies the value of an attribute for all objects in a subtree that have this attribute value. All objects of the subtree are checked and the number of hits are reported. You must have modify authorization for the attribute to be modified in the objects found.

The DN of the object must not contain alias names in its name parts. It is recommended that you use shadow administration functions to administer shadows (see Chapter 10). Only the master entries are changed by this function.

If the master entry to be modified is under a shadow entry, the DUA also automatically changes the relevant shadow entry under the master entry of the higher-level object. If this DSA is not available, the master entry is modified, and the shadow entry must be modified by the administrator when the DSA is made available.

Mask sequence

Mask 3	Select option number 4.
Mask 16	Select option number 7.
Mask 5	Select the structure rule of the root of the subtree. If you do not select **ROOT** as the structure rule, then Mask 6 is displayed.
Mask 6	Enter the DN of the root of the subtree.
Mask 6d	Select the attribute name of attribute to be modified.
Mask 8	If the attribute name is not a special attribute, Mask 8 is displayed for entering the old and new value. If the attribute name selected is a special attribute, the special masks are displayed twice; the old value must be entered the first time and the new value must be entered the second time. If the attribute name selected is **CDS-Replica** or **CDS-Cell**, or if the syntax of the attribute is TTX-ID, Telex Number syntax, Postal Address syntax, Fax Number syntax, MHS O/R Address syntax, MHS DL Submit Permission syntax or MHS O/R Name syntax then the following special masks are displayed.
Mask 21	Enter the old value of **CDS-Cell** first, followed by the new value.
Mask 22	Enter the old value of **CDS-Replica** first, followed by the new value.
Mask 23	Enter the old value of the attribute with TTX-ID syntax, followed by the new value.
Mask 24	Enter the old value of the attribute with Telex-Number syntax, followed by the new value.
Mask 25	Enter the old value of the attribute with Postal Address syntax, followed by the new value.
Mask 26	Enter the old value of the attribute with Fax Number syntax, followed by the new value.
Mask 27	Enter the old value of the attribute with the MHS O/R Address syntax, followed by the new value.

Depending on what you selected in the **O/R Address Type** field, one of the following masks will be displayed:

- Mask 28 for entering a mnemonic O/R address
- Mask 29 for entering a numeric O/R address
- Mask 30 for entering a structured postal O/R address
- Mask 31 for entering an unstructured postal O/R address
- Mask 32 for entering a terminal O/R address

Mask 33 Enter the old value of the attribute with the MHS DL Submit Permission syntax, followed by the new value.

If **Individual**, **Member of DL**, or **Pattern Match** is selected in the **DL Submit Permission Type** field, then Mask 34 is displayed; otherwise, Mask 35 (Member of Group) is displayed.

Mask 34 Enter the old value of the attribute with the MHS O/R Name syntax or MHS DL Submit Permission syntax followed by the new value.

The following attributes cannot be changed using the Modify Subtree operation:

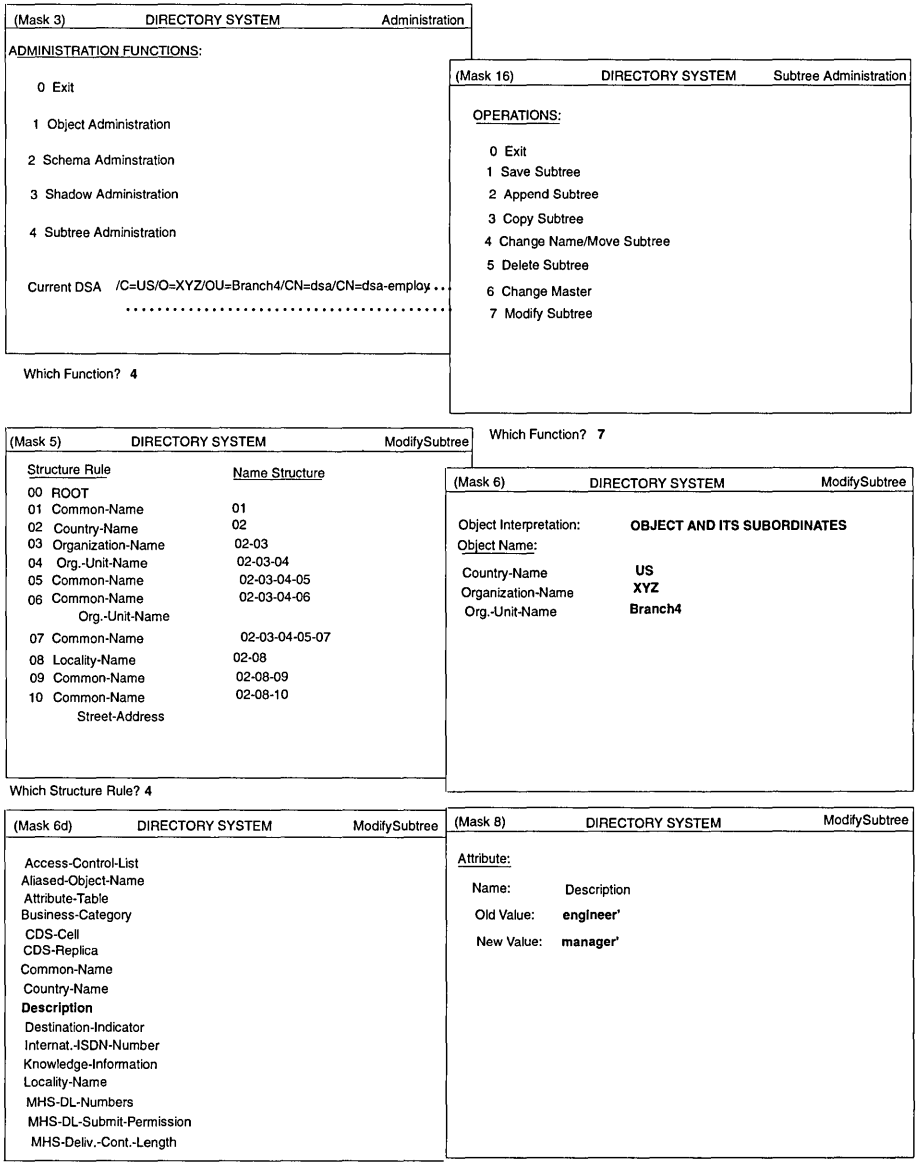
- **Presentation-Address**
- **Master Knowledge**

The same applies to special attributes with the following syntaxes:

- Preferred Delivery Method
- ASN1
- Any
- Search Guide

Figure 11-28 shows a sample mask sequence that modifies the **Description** attribute for all objects in the subtree of **/C=US/O=XYZ/OU=Branch4** from **engineer** to **manager**.

Figure 11–28. Sample Modify Subtree Operation



Part 2

DCE Distributed File Service

Chapter 12

An Overview of DFS

This chapter introduces basic concepts of the DCE Distributed File Service (DFS). It provides introductory information about the components of DFS, the administrative advantages they offer, and their interaction with other DCE components. It also provides a brief overview of some common DFS administrative tasks, explains DFS command structure, and describes how you can get help for DFS commands. You should read and understand this chapter before performing any of the tasks detailed in Part 2 of this guide.

12.1 Features of DFS

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

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

12.1.1 DFS Server Machines

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

Other server machine roles include: a System Control machine that updates other server machines with identical versions of system configuration files; Binary Distribution machines that distribute system binaries to other machines with the same CPU/operating system type; Fileset Database machines that house the master and replica versions of the Fileset Location Database (FLDB) where information about the location of system and user files is maintained; and Backup Database machines that house the master and replica versions of the Backup Database where information used to back up and restore system and user files resides. (See Chapter 13 for more information about the roles of DFS server machines.)

12.1.2 DFS Client Machines

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

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

When the Cache Manager receives requested data from a File Exporter, it caches the data (stores it on disk or in memory) before passing it to the application program that requested it. In addition, the Cache Manager ensures that the user is always working with the most current copy of the

file. If the central copy of the file changes, the Cache Manager is notified, and it retrieves a newer version of the file for the application program on the client machine the next time the file is referenced. The user does not have to direct the Cache Manager to keep a current copy; the Cache Manager's actions are automatic and completely transparent to the user.

12.1.3 DFS Data Access Management

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

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

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

12.1.4 DFS Administrative Domains

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

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

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

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

A cell can have one or more administrative domains. An administrative domain, like a cell, can include server machines that perform many of the machine roles mentioned previously. A machine can be a member of multiple domains, but all of the machines in a domain must be members of the same cell. For example, all of the domains in a cell can use the same

Binary Distribution machine for a machine type, but that machine must be in the same cell as all of the machines in all of the domains. Administrative domains are transparent from the end-user's perspective.

12.1.5 DFS Administrative Lists and Groups

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

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

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

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

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

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

12.1.6 DCE Local File System

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

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

DCE LFS aggregates also support the use of filesets. A DCE LFS fileset is a hierarchical grouping of files managed as a single unit. DCE LFS filesets can vary in size but are almost always smaller than a disk partition. In the DCE LFS, multiple filesets can be stored on a single aggregate, providing flexible disk usage. A non-LFS partition (for example, a UNIX partition) can be exported to the namespace for use as an aggregate with DFS. However, it can store only a single fileset (file system), regardless of the amount of data actually stored in the fileset. (The terms *non-LFS aggregate* and *non-LFS fileset* are used to refer to exported non-LFS partitions and the file systems they contain.)

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

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

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

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

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

12.1.7 DFS Replication

The DCE LFS allows you to replicate (copy) DCE LFS filesets. When you replicate a DCE LFS fileset, you place read-only copies of it on multiple server machines. The unavailability of a single server machine housing a replicated fileset does not interrupt work involving that fileset because copies of the fileset are still available from other machines. The replication of commonly used configuration and binary files on multiple server machines greatly reduces the chances of their being unavailable as the result of server machine outages. Replication also prevents a machine from becoming overburdened with requests for files from a frequently accessed DCE LFS fileset. Replication is supported only for DCE LFS filesets, not for non-LFS filesets (file systems on non-LFS partitions).

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

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

12.1.8 DFS Backup System

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

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

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

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

12.1.9 DFS Database Distribution

DFS houses fileset and Backup System information in two administrative databases: the Fileset Location Database (FLDB) stores information about the locations of filesets; the Backup Database records information about backups and tapes. To maintain file system reliability and availability, the

two databases are replicated on multiple server machines. If any of the machines housing a database becomes unavailable, the database is still available from other machines.

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

12.1.10 The DFS scout Program

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

The **scout** program tracks the following statistics about the File Exporter on each machine being monitored: the number of connections open between the File Exporter and client machines; the number of kilobytes of data that client machines have fetched from the File Exporter; the number of kilobytes of data that clients have sent to the File Exporter for storage; the number of client machines that have communicated with the File Exporter; and the number of kilobytes available on aggregates on the File Server machine.

12.2 Advantages of DFS

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

12.2.1 Faster Restarts and Better Reliability

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

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

DFS also improves the reliability of data access by allowing you to replicate commonly used DCE LFS filesets on multiple File Server machines. When you replicate a fileset, you place an identical copy of the fileset on a different File Server machine. The unavailability of a single server that houses the fileset generally does not interrupt work involving that fileset because the fileset is still available from other machines.

12.2.2 Better Recovery from Failure

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

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

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

12.2.3 Improved File Availability and Access Time

Increased availability of files in DFS is provided through two mechanisms: replication and caching. Replication increases file availability by allowing DCE LFS filesets to be replicated on multiple server machines; this minimizes the effects of machine outages. If one machine housing a read-only copy of a DCE LFS fileset is unavailable, other replicas of the fileset are still available from other machines.

Caching data locally decreases access time to the data. The cache is an area of a client machine's local disk or memory dedicated to temporary data storage. Once data is cached, subsequent access to it is fast because the client machine does not need to send a request for it across the network. Thus, caching also minimizes network traffic. As noted previously, DFS ensures that each client housing cached data always has access to the most-recent version of the data.

12.2.4 Efficient Load Balancing and File Location Transparency

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

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

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

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

12.2.5 Extended Permissions

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

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

12.2.6 Increased Interoperability and Scalability

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

In DFS, the Basic OverSeer Server (BOS Server) automatically monitors DFS processes on server machines. Once it is started and configured, the BOS Server continues to monitor other DFS server processes with minimal intervention from the system administrator. Decreased administrative

obligations, coupled with high performance and a high client to server ratio, make DFS a scalable system. Server and client machines can be added to a DFS configuration with little impact on other servers or clients and with few additional administrative responsibilities.

12.2.7 Increased Security and Administrative Flexibility

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

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

12.2.8 Consistency of Configuration and Binary Files

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

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

The DFS Update Server process distributes common files from System Control and Binary Distribution machines. Server machines that rely on

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

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

12.2.9 Backup Versions of Data

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

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

12.2.10 System Monitoring

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

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

12.3 Interaction with Other DCE Components

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

- The DCE Security Service, especially the keytab file and ACLs. (See the *OSF DCE Administration Guide—Core Components* for information about using keytab files, and the DCE Security Service portion of the *OSF DCE User's Guide and Reference* for complete details about setting ACLs.)
- The DCE Directory Service, especially details about the namespace. (See the *OSF DCE Administration Guide—Core Components* for complete details about configuring and using namespace components.)
- The DCE Distributed Time Service, especially client and server machine synchronization. (See the *OSF DCE Administration Guide—Core Components* for complete details about configuring and using the Distributed Time Service.)
- The DCE Remote Procedure Call facility, especially client and server machine communications. (See the *OSF DCE Administration Guide—Core Components* for complete details about configuring your machines to use RPCs.)

Many of these services also interact with each other. (See the *OSF DCE Administration Guide—Introduction* and the *OSF DCE Administration*

Guide—Core Components for more information on these components and their mutual interaction.)

12.3.1 DCE Security Service

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

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

These three services rely on the DCE Security Server, the **secd** process. The **secd** process runs on all DFS server machines to provide access to the security services in the previous list.

The DCE Security Service also includes the following two facilities:

- The DCE Access Control List Facility provides an interface that allows users to set different levels of protection on file system objects such as directories and files. Users can grant permissions to individuals, or they can define groups of users and grant permissions to the groups. They can then add individuals to a group to grant them the same permissions as the group, or they can remove individuals from a group to restrict their permissions. An object's ACLs interact with the protections provided by the object's UNIX mode bits.
- The DCE Login Facility initializes a user's security environment in DCE. It employs a user's password to authenticate the user to the DCE Security Service, returning authentication information associated with the user. This information is used to authenticate the user to other distributed services such as those in DFS.

12.3.2 DCE Directory Service

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

- The Cell Directory Service (CDS) manages names within a cell. Each resource has a CDS entry that is unique within its local cell.
- The Global Directory Service (GDS) supports the global naming environment between cells and outside of cells. GDS is an implementation of a directory services standard known as X.500.
- The Global Directory Agent (GDA) is a "gateway" between the local and global naming environments. It supports cell interoperability by allowing CDS to access a name in another cell via either GDS or the Domain Name System (DNS), a widely used global naming environment.

12.3.2.1 Examples of CDS Entries

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

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

The second example shows a similar entry in GDS format:

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

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

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

12.3.2.2 DFS Filespace

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

The name **fs** is only a default; it is not considered to be well known and, thus, can vary from cell to cell. (See the *OSF DCE Administration Guide—Introduction* for information on the DCE installation and configuration script that can be modified to specify a name other than **fs** as the junction to a cell's DFS filespace.)

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

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

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

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

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

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

The /.: and /: prefixes are abbreviations intended primarily for interactive use, not for use in persistent storage such as shell scripts. For example, suppose one of the abbreviations is included in a script. If the script is executed in the local cell, the abbreviation indicates the proper local

pathname. However, if the script is run in a foreign cell, the abbreviation does not indicate the proper pathname. The script is valid only within the cell in which it was written. If persistent storage may be used outside of the local cell, use global names, including the /... prefix. Note that the prefixes must be made available on a per-client machine basis.

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

12.3.3 DCE Distributed Time Service

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

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

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

12.3.4 DCE Remote Procedure Call

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

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

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

When a server process first starts, it registers its process endpoint with the **rpcd** process. The **rpcd** process running on a server machine provides the location information required by clients to communicate with server processes running on the server machine. Note that in DCE RPC, **rpccp** is a program that, among other things, allows system administrators to administer and control **rpcd** on a machine. Most RPC administrative tasks, however, are performed automatically when a server first starts.

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

12.4 System Administration: A Task Overview

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

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

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

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

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

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

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

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

system administrators to monitor File Exporter usage statistics. All 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. Section 12.5 provides information about the structure of DFS commands and describes how to receive online help for DFS commands.

Refer to the DCE DFS portion of the *OSF DCE User's Guide and Reference* and the DCE DFS portion of the *OSF DCE Administration Reference* for detailed discussions of the various DFS commands. Refer to the DCE Security Service portions of these documents for complete details about the security commands referenced in this section. Consult the remainder of the DFS chapters in this guide for information about fileset management, system management, and most security issues referenced in this section.

12.4.1 Fileset Management Commands

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

12.4.1.1 Fileset (fts) Commands

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

- Create a read/write fileset with the **fts create** command; mount the fileset in the file tree with the **fts crmount** command.
- Examine a mount point with the **fts lsmount** command; delete a mount point with the **fts delmount** command.
- Create a backup version of a single fileset with the **fts clone** command; create backup versions of many filesets at once with the **fts clonesys** command.
- Prepare to replicate a fileset by assigning replication parameters with the **fts setrepinfo** command.
- Define replication sites with the **fts addsite** command; remove replication sites and read-only replicas at the sites with the **fts rmsite** command.

- Create read-only replicas of a fileset with the **fts release** and **fts update** commands.
- Check the status of the Replication Server on a File Server machine with the **fts statrepserver** command; check the status of each replica of a fileset with the **fts lsreplicas** command.
- Set a fileset's quota with the **fts setquota** command; list the quota with the **fts lsquota** command.
- List fileset header information with the **fts lsheader** command.
- List FLDB information with the **fts lsfldb** command.
- Examine fileset header information and FLDB information with the **fts lsft** command.
- Move a fileset with the **fts move** command.
- Remove a fileset with the **fts delete** command.
- Dump a fileset to a byte stream format with the **fts dump** command; restore a fileset to the file system with the **fts restore** command.

12.4.1.2 Backup (bak) Commands

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

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

12.4.2 System Management and Configuration Commands

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

12.4.2.1 Cache Manager (cm) Commands

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

- List the current cache size and type with the **cm getcachesize** command.
- Set the size of the cache with the **cm setcachesize** command.
- Force the Cache Manager to discard cached data and information about the data with the **cm flush**, **cm flushfileset**, and **cm checkfilesets** commands.
- Determine whether the Cache Manager allows **setuid** programs from specific filesets to execute with **setuid** permission by using the **cm getsetuid** command.
- Allow or disallow **setuid** programs from specific filesets to execute with **setuid** permission by using the **cm setsetuid** command.
- Check the status of File Server machines with the **cm statservers** command.
- Determine the cell in which a file or directory is stored, the fileset in which it is stored, and the machine on which it resides with the **cm whereis** command.
- Determine Fileset Location Database machines for the local cell and any cells with which the Cache Manager has been in contact with the **cm lscellinfo** command.

12.4.2.2 Basic OverSeer (bos) Commands

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

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

12.4.2.3 The scout Program

You can use the **scout** program to monitor the following types of tasks:

- The number of connections a File Server machine has open to client machines
- The maximum number of kilobytes of data that clients have fetched from or sent to a File Server machine
- The number of active client machines a File Server machine is serving

- The number of kilobytes in use on each aggregate on a File Server machine

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

12.4.3 Security Commands and Tools

Commands in the **bos** suite are also used to manage DFS administrative privileges and security in a cell. In addition, DCE Security Service commands let you modify ACLs and groups.

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

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

You can also use the following DCE Security Service commands to perform security-related tasks:

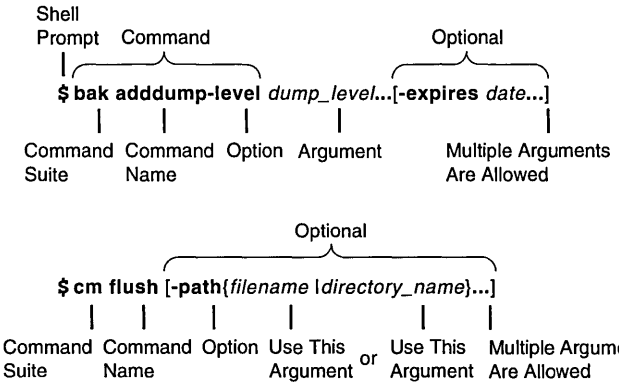
- Verify or modify ACL permissions with the **acl_edit** command.
- Create administrative (or user) groups by using the **rgy_edit** command.

12.5 DFS Command Structure and Help

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

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

The following examples illustrate the elements of a DFS command:



The following list summarizes the elements of a DFS command:

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

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

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

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

Arguments

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

Optional Information

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

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

12.5.1 Command Shortcuts

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

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax. (The syntax for each command is listed with its description in the DCE DFS portion of the

OSF DCE User's Guide and Reference or in the DCE DFS portion of 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 **-server** option found in many DFS commands can typically be omitted or abbreviated to be simply **-s**.

You can also abbreviate a command name to the shortest form that still distinguishes it from the other command names in its suite. For example, you can shorten the **fts help** command to **fts h** because no other command names in the **fts** command suite begin with the letter "h." However, there are several **fts** commands that begin with the letter "l," such as **fts lsquota**, **fts lsmount**, and others. To avoid ambiguity, you can abbreviate these commands to **fts lsq** and **fts lsm**; other **fts** command names that begin with "l" can be abbreviated in a similar fashion. Note that because miscellaneous DFS commands are not included in a suite, their names cannot be abbreviated.

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

- Complete command:

  $ fts lsquota -path jlw/doc jlw/public

- Abbreviated command name and abbreviated option:

  $ fts lsq -p jlw/doc jlw/public

- Abbreviated command name and omitted option:

  $ **fts lsq jlw/doc jlw/public**

12.5.2 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.

  $ **man** *command_suite*

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

  $ **man** *command_suite_command_name*

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

  $ *command_suite* **help**

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

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

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

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

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

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

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

Chapter 13

DFS Configuration Issues

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

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

13.1 Choosing DFS Machine Roles

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

Each DFS server or client machine must also run the RPC, CDS, and Security processes necessary for configuration as a DCE client machine. An RPC binding must be created in CDS for the DCE pathname of each server machine, and a DFS server principal must also be created in the Registry Database for each server machine. The following sections assume that these requirements have been satisfied prior to configuring a machine as a DFS server or client machine. (See the *OSF DCE Administration Guide—Introduction* for more information on fulfilling these requirements.)

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

Each DFS server process has an associated administrative list. Users, groups, and server machines included on a process's administrative list can issue commands or calls that affect the process. Members can be added to administrative lists at any time. Chapter 15 provides detailed information about the procedures used to create and maintain administrative lists. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about the administrative privileges and permissions required to issue each DFS command.)

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

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

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

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

13.1.1 Overview of DFS Machine Roles

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

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

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

- In a cell with only one server machine, the machine runs all processes and fills all the necessary machine roles. Note that the System Control machine and Binary Distribution machine roles are unnecessary in this configuration.
- In a cell with two server machines, both machines act as Fileset Database machines and Backup Database machines to replicate the databases. For each database, one of the machines automatically assumes the role of the synchronization site and houses the source copy of the database. If one of the machines becomes unavailable, the information in the database may not be able to be changed. (See Section 13.5 for a detailed description of database synchronization.)
- In a cell with three or more server machines, three machines run as Fileset Database machines and three machines run as Backup Database machines. This configuration allows the cell to benefit from the database replication capabilities of DFS. An odd number of database machines is best.

The software for all server processes can be installed on every server machine, even though a machine need not run every process. To then change the role of a machine, simply start or stop the appropriate processes. Machines roles are not mutually exclusive: any server machine can assume multiple server machine roles, any server machine can be configured as a client machine, and any client machine can be configured as a server machine.

13.1.1.1 System Control Machines

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

The following processes run on a System Control machine:

- An **upserver** process (the server portion of the Update Server), which controls the distribution of common configuration files to all other server machines in the domain.
- An **upclient** process (the client portion of the Update Server), which retrieves binary files from the Binary Distribution machine of the proper CPU/OS type. (See Section 13.1.1.2 for a description of the Binary Distribution machine.)
- A BOS Server (**bosserver** process). (See Section 13.1 for more information about the BOS Server.)

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

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

13.1.1.2 Binary Distribution Machines

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

Configure one Binary Distribution machine for each CPU/OS type for which multiple machines exist in the cell.

A Binary Distribution machine runs the following processes:

- An **upserver** process (the server portion of the Update Server), which controls the distribution of binary files to other server machines of the same CPU/OS type in the cell.
- An **upclient** process (the client portion of the Update Server), which retrieves configuration files from the System Control machine.
- A BOS Server (**bosserver** process). (See Section 13.1 for more information about the BOS Server.)

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

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

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

13.1.1.3 File Server Machines

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

- A Fileset Server (**ftserver** process).
- The File Exporter, which is initialized by the **fxd** process, in the kernel.
- The **dfsbind** process.
- The Replication Server (**repserver** process).
- Two **upclient** processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server (**bosserver** process). (See Section 13.1 for more information about the BOS Server.)

The Fileset Server, or **ftserver** process, provides an interface for commands that affect filesets (commands that create, delete, or move filesets, and commands that prepare filesets for archiving to tape or other media). The most common occurrences of fileset creation and deletion are when you add or remove users from the system. Filesets are most often moved to provide load balancing among File Server machines.

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

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

- Delivering requested files and programs to clients; storing files and programs when clients finish with them
- Maintaining the directory hierarchy structure

- Handling file-related or directory-related requests (creating, deleting, copying, and moving filesets)
- Tracking status information (including size and modification status) about each file and directory
- Creating symbolic links between files

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

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

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

- *When accessing a file or directory via its DCE pathname*, if the user is logged into the local machine as **root** but is not authenticated to DCE, DFS treats **root** as an unauthenticated user; the **root** user has no special privileges. If the user is also authenticated to DCE as **root**, DFS treats the user according to **root**'s DCE identity.

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

- *When accessing a file or directory via its local pathname*, the **root** user has all of the privileges commonly associated with **root**. For local

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

Being a member of the **fxd** administrative group allows you to perform any operation on a file or directory in an exported fileset, but you may have to change the file's or directory's protections first. Being logged into the local machine as **root** lets you perform any operation on a file or directory in a locally mounted fileset immediately.

The File Exporter also manages the distribution of tokens to clients. It maintains an inventory of outstanding tokens, including the clients to which it has granted tokens, the data for which it has granted those tokens, and the type of each token it has granted. (A token's type dictates the operations that the client holding the token can perform on the data to which the token applies.) (See Section 13.4 for more information about the File Exporter's token-management mechanism.)

The **fxd** process must be run on any machine used to export data to the global namespace. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about the **fxd** process.)

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

The **dfsbind** process must also be run on all client machines. Its role on client machines is described along with client machines and their processes in Section 13.2.2. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about the **dfsbind** process.)

The Replication Server, or **repserver** process, manages replicas of filesets on all File Server machines. Depending on the replication method in use, you either release a new version of a fileset for distribution by the Replication Server, or the Replication Server automatically creates replicas

at specified intervals. Install the Replication Server on all File Server machines, which are the machines that can store read-only replicas of filesets. No administrative list is associated with the **repserver** process.

In addition, each File Server machine must have a server entry registered in the FLDB before it can house filesets. The server entry must exist before the **fts create** or **fts crfldbentry** command can be used to create an entry in the FLDB for a DCE LFS or non-LFS fileset from the machine. (See Chapter 17 for more information about creating server entries.)

A client machine can also be configured as a Private File Server machine to export data to the global namespace. (See Section 13.1.1.7 for more information about configuring a client machine to export data.)

13.1.1.4 Fileset Database Machines

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

Each Fileset Database machine runs the following processes:

- A Fileset Location Server (**flserver** process).
- Two **upclient** processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server (**bosserver** process). (See Section 13.1 for more information about the BOS Server.)

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

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

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

A user can issue commands that affect FLDB entries for filesets on a server machine without being listed in the **admin.fl** list, provided the user owns the machine's server entry in the FLDB. A user gains ownership of a server entry in the FLDB by being included in the group specified as the owner of that machine's entry with the **fts crserverentry** command. (See Chapter 17 for more information about creating server entries in the FLDB.)

13.1.1.5 Backup Database Machines

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

Each Backup Database machine runs the following processes:

- A Backup Server (**bakserver** process).
- Two **upclient** processes: one to retrieve configuration files from the System Control machine, and one to retrieve binary files from the Binary Distribution machine of the proper CPU/OS type.
- A BOS Server (**bosserver** process). (See Section 13.1 for more information about the BOS Server.)

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

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

Commands in the **bak** suite are used to communicate with the DFS Backup System. They can be entered from any machine in the cell. Data is physically backed up and restored on a Tape Coordinator machine, which is a client or server machine that has a tape drive and runs the **butc** process to manage the drive. Information stored in the Backup Database determines the data to be backed up by a Tape Coordinator machine. (See Chapter 20 for more information on configuring and using Tape Coordinator machines.)

13.1.1.6 DFS Client Machines

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

Each client machine must run

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

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

When the Cache Manager receives the requested file, it stores the file in its local cache, which is an area reserved for data storage on disk or in memory on the client machine. It then passes the file to the application program. The Cache Manager also stores tokens it receives from the File Exporter on the File Server machine. It uses the tokens to track the state of the data in its cache as compared to the state of the central copy of the data at the File Server machine. The cached data remains current for as long as the Cache Manager's tokens remain valid. If the central copy of the file changes, the File Exporter revokes the tokens. The next time the data is requested, the Cache Manager retrieves the newer version to its cache before providing it to the application program.

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

The **dfsbind** process, in addition to its role on File Server machines, is used by the Cache Managers on client machines to help with the resolution of DCE pathnames. It also obtains authentication information about users that Cache Managers require for RPC bindings to File Server machines. (See Section 13.2.2 for more information about the Cache Manager and the **dfsd** and **dfsbind** processes.)

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

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

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

Although meeting these requirements qualifies a client machine as a File Server machine, that is not the machine's primary role. The machine's local disk is not to be used for data storage for an entire cell or domain. A client machine meets the previous requirements solely to allow users who administer the machine to make data on its local disk available in the global

namespace. (See Section 13.1.1.3 for more information about these additional processes.)

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

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

13.1.2 Summary of DFS Machine Roles

Table 13-1 summarizes the DFS machine roles described in the previous sections. For each machine role, the table provides a brief description of its purpose and lists the DFS processes that a machine filling the role must run. The table also provides suggestions for how to configure machines of a specific type and other roles a machine of each type can assume. A machine that is assuming any of the roles listed in the table must be configured as a DCE client machine. A machine assuming a role as a DFS server must have both an RPC binding in CDS for its pathname and a DFS server principal in the Registry Database.

Recall that any server machine can be configured to perform any of the other server machine roles. Also, a server machine can be configured as a client machine, and vice versa. For a machine to fill an additional role, it must run the processes listed for that role in the third column of the table. (See Section 13.1.1 for expanded descriptions of the machine roles.)

Note: Table 13-1 uses the numbers **1** and **2** to differentiate the **upserver** and **upclient** processes running on the machines. The notations **upserver**[1] and **upclient**[1] denote the Update Server that distributes common configuration files from a System Control machine. The notations **upserver**[2] and **upclient**[2] denote the Update Server that distributes binary files from a Binary Distribution machine.

Table 13–1. Summary of DFS Machine Roles

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

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

13.2 DFS Server and Client Configuration Issues

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

13.2.1 Server Machine Processes and Files

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

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

Several directories contain files related to DFS server processes. The directories in the following list store files on a server machine's local disk. Files stored on the local disk are generally required for DFS to start without accessing the global namespace. (See the *OSF DCE Administration Guide—Introduction* for complete details about the files that must reside on the local disk of a server machine.)

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

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

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

13.2.2 Client Machine Processes and Files

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

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

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

In addition, the **dfsd** process starts several background daemons. These daemons include one or more maintenance daemons that perform routine maintenance tasks such as garbage collection; background daemons that improve performance by performing delayed writing of updated data; token daemons that respond to token revocation requests from File Exporters; and, on the AIX operating system, I/O daemons that move data between disk and memory.

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

The **dfsbind** process also returns user authentication information from the Security Server to the kernel RPC Runtime of the client machine. Authentication information must be included in RPC bindings that request data from a File Server machine for a user. The Cache Manager uses the RPC bindings to access data for the user from the File Server machine. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about the **dfsd** and **dfsbind** commands that start the respective processes and the options available with the commands.)

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

During a reboot, DFS is inaccessible until the **dfsd** process reinitializes the Cache Manager; the **dfsbind** process must be running before the **dfsd** process can be run. Any files that are needed during reboot prior to the start of the **dfsd** process must reside on the local disk. Following is a list of recommended DFS files to store on a local disk. (See the *OSF DCE Administration Guide—Introduction* for complete details about all of the files that are needed on the local disk of a client machine.)

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

You may also wish to store diagnostic and recovery files on a local disk. Certain commands in the **bos** and **cm** command suites can help users diagnose problems caused by a File Server outage. It is useful to have local

disk copies of the binary files for the **bos** and **cm** suites because the File Server outage that requires their use can also make them inaccessible. In addition, you may wish to keep the binaries for a text editor, such as **ed** or **vi**, on the local disk for use during outages.

13.3 Setting Up Filesets

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

The following subsections discuss setting up a cell's root fileset, binary and configuration filesets, and user filesets. Information about fileset replication and the **@sys** and **@host** variables, which simplify cell administration, is also provided. (See Chapter 17 for complete information about creating and mounting filesets; see the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about **fts** and other DFS commands.)

13.3.1 Setting Up the Root Fileset

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

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

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

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

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

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

Once these steps are complete, you can replicate **root.dfs**. Replication is then available for DCE LFS filesets created in the cell. It is important that you follow these instructions if you plan to replicate filesets in your cell. Due to the nature of mount points, if you replicate **root.dfs** before creating its read/write mount point, you effectively make it impossible to access the read/write version of **root.dfs**. (See Chapter 17 for more information about creating and mounting filesets, using mount points, and creating and exporting aggregates; see the *OSF DCE Administration Guide—Introduction* for instructions on configuring your root fileset to support replication.)

Note: By default, the junction to the DFS filespace is defined at /.../*cellname*/**fs**. However, the name of the junction is not considered to be well known and can be changed by modifying the DCE installation and configuration script used to install and configure the machines in a DCE cell. (See the *OSF DCE Administration Guide—Introduction* for more information about the installation and configuration script.) The examples in this part of the guide use the default, **fs**, as the junction of the DFS filespace.

13.3.2 Choosing Fileset Names

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

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

There is a 111-character limit on the length of fileset names. However, because a **.readonly** extension is added when you replicate a fileset, you need to specify fileset names that contain no more than 102 characters. When creating filesets, do not add the **.readonly** and **.backup** extensions yourself; DFS automatically adds the appropriate extension.

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

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

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

- Use the **common.***type* prefix for common filesets. For example, use **common.etc** for common configuration files (mounted at /.../*cellname*/**fs/common/etc**), and **common.forms** for common forms (mounted at /.../*cellname*/**fs/common/forms**).
- Use the **src.***type* prefix for source filesets. For example, use **src.dfs** for DFS source files (mounted at /.../*cellname*/**fs/src/dfs**).
- Use the **user.***username* prefix for all user filesets. For example, use **user.terry** for user **terry**'s fileset (mounted at /.../*cellname*/**fs/usr/terry**).
- Use the **public.***username* prefix for each user's public fileset. For example, use **public.terry** for **terry**'s public fileset, which contains information the user wants to make available to everyone. The **public.terry** fileset is mounted at /.../*cellname*/**fs/public/terry**.
- Use the *sys_type.distribution_dir* prefix for operating system-specific filesets. For example, use **osf1_pmax.bin** for OSF/1 binary files (mounted at /.../*cellname*/**fs/osf1_pmax/bin**), and **osf1_pmax.lib** for OSF/1 libraries (mounted at /.../*cellname*/**fs/osf1_pmax/lib**). In DFS,

symbolic links are often created from the **/bin** and **/lib** directories (or their equivalents) on the local disk of a workstation to these DFS mount points.

(See Chapter 17 for more information on additional rules for naming filesets.)

13.3.3 Setting Up Binary and Configuration Filesets

You may find it convenient to store DCE binaries, system binaries, and configuration files (for example, those commonly found in directories such as **/bin** and **/etc** or their equivalents) in the DFS filespace, instead of on the local disk of each machine. Because binary files are operating system specific, you may want to create a different fileset for each system type (for example, **osf1_pmax** or **aix32_rs**) and distribution directory (for example, **/etc** and **/bin**) and store the filesets on a DFS File Server machine. You can then create symbolic links from the local disk to the fileset.

Note that DFS simplifies the creation of such links by providing the **@sys** variable, which is set on a per-Cache Manager basis. When the Cache Manager encounters the **@sys** variable in a pathname, it substitutes its system name for the variable. (See Section 13.3.7 for a more detailed description of the **@sys** variable.)

For example, while it is a good practice to store the binary files for a single text editor on the local machine, the binaries for other text editors do not need to be stored on each machine. A system administrator can create filesets that store text editor binaries for each system type. The administrator can then construct a symbolic link from the local disk of each machine to the appropriate fileset in DFS. For instance, system administrators in the **abc.com** cell, which runs the OSF/1 and AIX 3.2 operating systems, can configure part of their file tree as shown in Table 13-2.

Table 13–2. Examples of Fileset Names and Mount Points for Binary Files

Fileset Name	Mount Point
osf1_pmax	**/.../abc.com/fs/osf1_pmax**
osf1_pmax.bin	**/.../abc.com/fs/osf1_pmax/bin**
osf1_pmax.etc	**/.../abc.com/fs/osf1_pmax/etc**
aix32_rs	**/.../abc.com/fs/aix32_rs**
aix32_rs.bin	**/.../abc.com/fs/aix32_rs/bin**
aix32_rs.etc	**/.../abc.com/fs/aix32_rs/etc**

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

13.3.4 Setting Up User Filesets

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

As with any other fileset, you may want to create additional filesets based on logical file groupings and mount them below */.../cellname*/**fs/usr**/*username* if the user's fileset becomes too large. For example, if **terry** has 5000 kilobytes of data in the **project1** subdirectory and 3000 kilobytes of data in the **project2** subdirectory, you may want to create two smaller filesets organized below **/.../abc.com/fs/usr/terry**. Table 13-3 lists the organization and names of the filesets in this example.

Table 13–3. Examples of Fileset Names and Mount Points for User Data

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

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

The guidelines in Section 13.3.4 assume that the user does not have an existing home directory in the file system. A user who has data in an existing home directory in a non-LFS fileset mounted in the global namespace can continue to use that fileset. However, if you choose to create and mount a DCE LFS fileset for the user, you must be careful; DFS does not allow you to mount a fileset at an existing directory. You must move the user's data from the existing home directory in the DCE namespace to a temporary directory. You must then remove the existing home directory before creating and mounting the user's DCE LFS fileset. You can then move the user's data to the new fileset.

For example, suppose the user named **terry** in the previous example has an existing home directory in a non-LFS fileset mounted at **/.../abc.com/fs/usr/terry**. In this case, move the user's data from **/.../abc.com/fs/usr/terry** to a temporary location (such as a subdirectory of **/tmp** on the local disk), and remove the **/.../abc.com/fs/usr/terry** directory and its contents. Then, create and mount the user's DCE LFS fileset as described in Section 13.3.4. Finally, move the user's data from the temporary location into the new DCE LFS fileset mounted at **/.../abc.com/fs/usr/terry**. When these steps are complete, the user can access the data as before.

13.3.6 Replicating DCE LFS Filesets

You replicate DCE LFS filesets by placing read-only copies of them on one or more File Server machines in a cell. If a machine that houses a read-only copy of the fileset becomes unavailable, the information is still available from a copy of the fileset on another machine.

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

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

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

13.3.7 Using the @sys and @host Variables

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

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

13.3.7.1 The @sys Variable

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

On a machine running OSF/1:

```
$ cm sysname

Current sysname is 'osf1_pmax'

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

/.../abc.com/fs/osf1_pmax
```

On a machine running AIX 3.2:

```
$ cm sysname

Current sysname is 'aix32_rs'

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

/.../abc.com/fs/aix32_rs
```

The **@sys** variable is commonly used in symbolic links from a DFS client machine to a fileset in the DFS filespace. A single copy of a binary file for each system type is stored on a single File Server machine in DFS instead

of on the local disk of each client machine. Links are then created from client machines to the central copy of the binary file, eliminating the need to store the same binary file on each client machine. Accessing binary files this way saves disk space on client machines and ensures that users on all client machines are using the same version of the binary file. It also eases system administration by allowing administrators to update central copies of binary files, rather than requiring them to update the copies stored on each client machine.

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

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

On a machine running OSF/1:

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

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

$ cd /usr/local
$ pwd

/.../abc.com/fs/osf1_pmax/usr/local
```

On a machine running AIX 3.2:

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

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

$ cd /usr/local
$ pwd

/.../abc.com/fs/aix32_rs/usr/local
```

When creating links on server machines, do not use links to access binary files for DFS server processes. These files must reside on the local disk of each server machine to avoid bootstrapping problems. (See the DCE DFS portion of the *OSF DCE Administration Reference* for more information about the **cm sysname** command.)

13.3.7.2 The @host Variable

The **@host** variable is expanded to the value defined by the **hostname** command (or its equivalent) of the local operating system. The **@host** variable is especially useful when configuring machines that must execute a machine-specific set of start-up routines. For this reason, the **@host** functionality is most frequently used in cells that support diskless machines.

For example, all diskless machines in a cell could have symbolic links from the proper initialization file (**/etc/rc** or its equivalent) to **/.../abc.com/fs/diskless_config/@host/rc**. A diskless machine with a hostname defined as **diskless1.abc.com** would execute the file named

/.../abc.com/fs/diskless_config/diskless1.abc.com/rc

A diskless machine with a hostname of **diskless2.abc.com** would execute the file named

/.../abc.com/fs/diskless_config/diskless2.abc.com/rc

13.4 Data Access Management in DFS

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

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

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

13.4.1 Tokens

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

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

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

Open Tokens	Allow a client to open an entire file or fileset to read from it, write to it, delete it, or prevent it from being deleted. For example, a client that wants to open a file for reading requests an open token for the file.
Status Tokens	Allow a client to read or write file status information. For example, a client that wants to append data to a file, thus changing its size, needs a status token for the file.
Data Tokens	Allow a client to read from or write to a range of bytes in a file. For example, a client that wants to modify the first 10 bytes of a file requests a data token for those bytes.
Lock Tokens	Allow a client to read lock or write lock a range of bytes in a file. For example, a client that must ensure that only one process is locking the first 10 bytes of a file requests a lock token for those bytes.

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

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

13.4.2 Token Management

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

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

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

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

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

needs the client's permission to revoke tokens that are held by the client as long as the client's host lifetime has not expired.

Host RPC Lifetime

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

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

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

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

13.4.3 Token State Recovery

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

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

During the first form of token state recovery, the File Exporter attempts to preserve the state of its tokens across restarts by initially accepting requests only to reestablish existing tokens. While the File Exporter is unavailable, clients that have tokens from it continue to probe it at regular polling intervals until it returns to service. When it is again available, the File Exporter enters token state recovery to give these clients the opportunity to recover their tokens without threat of conflicts with tokens that were granted to new clients.

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

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

Values that the File Exporter uses to determine the host lifetimes, host RPC lifetimes, and polling intervals of its clients are specified with options of the **fxd** command. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about the **fxd** command and its options.)

13.5 DFS Distributed Database Technology

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

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

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

13.5.1 Ubik Database Synchronization

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

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

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

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

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

The coordinator at the synchronization site constantly tallies the votes it receives from the secondary sites. It continues in its role as synchronization site, confident that the other sites have not chosen a new synchronization

site and begun making competing changes to the database, as long as it receives the votes of a strict majority (more than 50%) of all database sites, including itself. The necessary majority of database sites is referred to as a quorum.

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

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

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

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

Immediately following the election, the newly elected synchronization site polls all sites to find the database with the highest version number. It adopts this version as the master copy and distributes it to the sites that do not yet have it. During the election and distribution, the database is unavailable.

13.5.2 Providing Information for Ubik

For the most part, Ubik operates without human intervention. However, it does depend on other DCE facilities and services for some things. The following list describes the interaction between Ubik and the remainder of DCE. It also provides an overview of the configuration information necessary for Ubik to operate properly. Section 13.5.3 discusses the database server configuration steps required for Ubik to function properly.

- *Ubik relies on DTS* to synchronize the clocks on server machines that house copies of a replicated database. Ubik coordinators must agree on the time; clock differences among Ubik sites can cause them to believe they are no longer in contact with each other, even if they are operating correctly. If a site falls out of touch, it may try to elect a new synchronization site or refuse to give out information. You can prevent such service outages by using DTS to synchronize the clocks on database server machines.

- *Ubik relies on the Security Service* for secure communications between all Fileset Database machines (machines that house the FLDB) and Backup Database machines (machines that house the Backup Database). Each type of database server has its own security group, of which all machines that house a copy of that type of database must be members. A machine's membership in this group allows the Ubik coordinator on that machine to communicate with the Ubik coordinators on the other database servers of that type.

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

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

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

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

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

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

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

13.5.3 Configuring Database Server Machines for Ubik

A cell's initial database server machines are configured when DFS is installed and configured in the cell. If it becomes necessary to add or remove database server machines after initial cell configuration, perform the steps in Sections 13.5.3.1 and 13.5.3.2 to properly configure information for Ubik. (You may be able to use the DCE installation and configuration script to modify the database servers configured in your cell. See the *OSF*

DCE Administration Guide—Introduction for more information about the script.)

When the Cache Manager on a DFS client machine needs information from the FLDB in a cell, the **dfsbind** process on the machine provides it with information about the names and network addresses of the Fileset Database machines for the cell. The information is valid for a limited amount of time, 24 hours by default, at which time the Cache Manager requests refreshed information from **dfsbind**; the Cache Manager also needs to refresh the information when it is restarted. The **fxd** process on a File Server machine passes the same information about Fileset Database machines for the local cell to the File Exporter on its machine, but only when it is restarted (generally when the machine is rebooted).

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

Sections 13.5.3.1 and 13.5.3.2 describe the steps required to add or remove a database server machine after initial cell configuration. Recall that each Fileset Database machine must run the FL Server (**flserver** process), and each Backup Database machine must run the Backup Server (**bakserver** process). These processes should be controlled by the Basic OverSeer (BOS) Server (**bosserver** process) on their machines, as recommended; if they are, you can use the appropriate **bos** commands to manipulate them.

Also recall that each FL Server must use the same **admin.fl** list, and that each Backup Server must use the same **admin.bak** list. The Update Server should be used to propagate these administrative lists from the System Control machine to their respective database server machines.

Refer to the sections at the beginning of this chapter for more information about the processes that must run on either type of database server machine and the administrative lists used to specify who can control them. (See Chapter 16 for more information about **bos** commands.)

13.5.3.1 Adding a Database Server Machine

To add a database server machine, do the following:

1. Use the **rgy_edit member -a** command in the group domain to add the abbreviated DFS server principal of the new database server machine to the appropriate security group (**subsys/dce/dfs-fs-servers** or **subsys/dce/dfs-bak-servers**) in the Registry Database.
2. Use the **rpccp add member** command to add the name of the RPC binding of the new database server machine to the appropriate RPC server group (*/.../cellname*/**fs** or */.../cellname*/**subsys/dce/dfs/bak**) in CDS.
3. Copy the appropriate administrative list (**admin.fl** or **admin.bak**) to the *dcelocal*/**var/dfs** directory on the new database server machine. These administrative lists are typically propagated from the cell's System Control machine via the Update Server.
4. Use the **rpccp show entry -u** command on each database server machine of the appropriate type to update the entry for the appropriate RPC server group from CDS. The command forces CDS to update information that it caches from the entry for the group in the namespace.
5. Stop and restart the appropriate database server process (**flserver** or **bakserver**) on each database server machine of that type. Restarting the existing database server processes causes the processes to read the updated RPC server group. This ensures that each Ubik coordinator agrees on the number and identities of the other database server machines of its type, which is vital to Ubik's use of a quorum of database server machines to maintain database consistency.
6. Start the appropriate database server process (**flserver** or **bakserver**) on the new database server machine.

13.5.3.2 Removing a Database Server Machine

To remove a database server machine, do the following:

1. Stop the appropriate database server process (**flserver** or **bakserver**) on the database server machine to be removed.
2. Use the **rgy_edit member -r** command in the group domain to remove the abbreviated DFS server principal of the database server machine to be removed from the appropriate security group.
3. Use the **rpccp remove member** command to remove the reference to the RPC binding of the database server machine to be removed from the appropriate RPC server group.
4. Use the **rpccp show entry -u** command on each database server machine of the appropriate type to update the entry for the appropriate RPC server group from CDS. The command forces CDS to update information that it caches from the entry for the group in the namespace.
5. Stop and restart the appropriate database server process (**flserver** or **bakserver**) on each database server machine of that type. Restarting the existing database server processes causes the processes to read the updated RPC server group. This ensures that each Ubik coordinator agrees on the number and identities of the other database server machines of its type, which is vital to Ubik's use of a quorum of database server machines to maintain database consistency.
6. Remove the appropriate administrative list (**admin.fl** or **admin.bak**) from the *dcelocal*/**var/dfs** directory on the database server machine to be removed. Modify the Update Server as necessary if the list is propagated from the cell's System Control machine.

Chapter 14

Using ACLs and Groups

This chapter summarizes the use of DCE Access Control Lists (ACLs) with DFS. DCE ACLs allow you to specify access to files and directories for individuals and groups of users. DCE ACLs can be used to protect files and directories stored in DCE LFS filesets. (See the DCE Security Service portion of the *OSF DCE User's Guide and Reference* for details about manipulating DCE ACLs.)

This chapter also presents information about groups. In addition to using groups in ACLs, you can use groups in DFS administrative lists to specify the users who are allowed to issue commands that affect filesets and server processes. In this manner, you can precisely control the security of the administrative domains in your cell. (See Chapter 15 for complete details about using administrative lists; see the *OSF DCE Administration Guide—Core Components* for information about creating and maintaining groups.)

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

14.1 Using DCE ACLs with DFS

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

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

DCE ACLs supplement the UNIX mode bits that are used to protect files and directories in DCE LFS filesets; they do not replace them. DCE LFS ensures that an object's mode bits and its ACL permissions are always synchronized. Note that objects in DCE LFS filesets can rely exclusively on mode bits as their sole form of protection. (See Sections 14.1.5 and 14.1.6 for more information about this possibility; see Section 14.1.4 for a description of the interaction and level of compatibility between DCE ACLs and UNIX mode bits.)

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

14.1.1 ACL Entries

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

type[:*key*]:*permissions*

The elements of an entry provide the following information:

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

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

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

14.1.1.1 ACL Entry Types for Users and Groups

DCE ACLs can include a number of different entry types, with each type specifying the permissions that are granted to different users. Some types require that a key be used to specify a user or group; others do not accept a key. Table 14-1 lists the different types of ACL entries, their use of entry keys, and the users and groups to which they apply.

Table 14–1. ACL Entry Types for Users and Groups

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

Some examples of ACL entries for users and groups follow:

user_obj:*permissions*

Defines the permissions for the user who owns the object.

user:jlw:*permissions*

Defines the permissions for a user named **jlw** from the local cell.

group:writers:*permissions*

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

foreign_user:/.../abc.com/wvh:*permissions*

Defines the permissions for a user named **wvh** from the foreign cell named **abc.com**.

foreign_group:/.../abc.com/alpine:*permissions*

Defines the permissions for a group named **alpine** from the foreign cell named **abc.com**.

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

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

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

14.1.1.2 ACL Entry Types for Masks

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

mask_obj:*permissions*

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

The **mask_obj** entry can only restrict the permissions granted by another entry; it cannot extend them. When DCE LFS determines the permissions

granted to a user by an entry to which the **mask_obj** applies, it compares the permissions granted by the applicable entry with those permitted by the **mask_obj** entry. DCE LFS denies the user a permission granted by the applicable entry if the permission is not included in the permission set specified with the **mask_obj** entry. DCE LFS does not grant the user a permission specified with the **mask_obj** entry but not with the applicable entry.

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

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

14.1.1.3 ACL Entry Types for Unauthenticated Users

Note: DCE ACLs used with objects for other DCE components include an additional **unauthenticated** entry type that serves as a mask for unauthenticated users. With other DCE components, when a user attempts to access an object, the system first determines the user's permissions and then determines whether the user is authenticated; if the user is not authenticated, the user's permissions are masked by the **unauthenticated** entry. While the **unauthenticated** entry can be included on the ACL of a DCE LFS object, DCE LFS ignores it.

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

When a user attempts to access an object, DFS first determines whether the user is authenticated. For access to an object in a DCE LFS fileset, an authenticated user acquires the permissions associated with the user's authenticated identity according to the normal ACL evaluation routine. (See Section 14.1.2 for a description of ACL evaluation.) An unauthenticated user's permissions are determined as follows:

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

   DFS assigns the identity **nobody** to all unauthenticated users, treating the identity as an authenticated user from an unknown foreign cell, regardless of the cell from which an unauthenticated user requests access to an object. DFS uses a fictitious cell as the local cell of the identity **nobody**; an entry for the fabricated cell cannot be created on an ACL. The user ID and group ID of the identity **nobody** are both typically **-2**, but they can vary between File Server machines.

2. DCE LFS grants the user the permissions associated with the **any_other** entry.

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

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

14.1.1.4 ACL Permissions

Each ACL entry for a user or group includes a set of permissions to define the operations it grants to the user or users to whom it applies. For a mask entry, the permissions define the maximum set of permissions that are allowed by the mask. Each entry can be assigned a different set of permissions. The following permissions can be associated with an entry on an ACL for a file or directory in a DCE LFS fileset. All six permissions apply to a directory, but only the first four apply to a file; the insert and delete permissions are meaningless for files.

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

Table 14-2 lists the various operations that can be performed on a file or directory and the ACL permissions that are required to perform them. As the table indicates, all operations performed on a file or directory object require the **x** (execute) permission on each directory that leads to the object. Keep this requirement in mind when determining the permissions necessary to perform the operations described in the following chapters; not all operations explicitly list it.

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

Table 14–2. File and Directory Operations and Required ACL Permissions

Operation	Required Permissions
Change to a directory	**x** on the directory itself **x** on all directories that lead to the directory
List the contents of a directory	**r** on the directory itself **x** on all directories that lead to the directory
List information about the objects in a directory	**r** and **x** on the directory itself **x** on all directories that lead to the directory
Create an object	**w**, **x**, and **i** on the directory in which the object is to be placed **x** on all directories that lead to the directory in which the object is to be placed
Delete an object	**w**, **x**, and **d** on the directory from which the object is to be deleted **x** on all directories that lead to the directory from which the object is to be deleted
Rename an object	**w**, **x**, and **d** on the object's current directory **x** on all directories that lead to the object's current directory **w**, **x**, and **i** on the object's new directory **x** on all directories that lead to the object's new directory
Read or read lock a file	**r** on the file itself **x** on all directories that lead to the file
Write or write lock a file	**w** on the file itself **x** on all directories that lead to the file

Operation	Required Permissions
Execute a binary file	**x** on the file itself **x** on all directories that lead to the file
Execute a shell script	**r** and **x** on the script itself **x** on all directories that lead to the script
List the ACLs on an object	**x** on all directories that lead to the object
Change the ACLs on an object	**c** on the object itself **x** on all directories that lead to the object

Note: In Table 14-2, the operation "List the contents of a directory" refers to displaying a simple list of the objects in a directory (for example, using the UNIX **ls** command with no flags or using **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).

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

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

```
user:rajesh:-wx-i-
```

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

```
user:rajesh:--x---
```

14.1.2 ACL Evaluation

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

To determine a user's permissions for an object, DCE LFS evaluates the entries on the object's ACL in a specific order. The following evaluation sequence defines the order in which DCE LFS checks ACL entries. ACL evaluation stops once the user matches the conditions in one of the steps in the evaluation sequence. Therefore, it proceeds to the next step in the sequence only if the user failed to match the conditions in all of the previous steps. (See Table 14-1 for a description of the different ACL entry types referred to in the following list.)

1. The user matches the **user_obj** entry. If the **user_obj** entry applies, DCE LFS grants the user the permissions specified in the entry. Note that the **user_obj** entry always explicitly has the **c** permission; the **c** permission cannot be removed from the **user_obj** entry.
2. The user matches a **user** or **foreign_user** entry. If a **user** or **foreign_user** entry on the ACL applies, DCE LFS grants the user the specified permissions after filtering them through the **mask_obj**.
3. The user matches one or more **group_obj**, **group**, or **foreign_group** entries. If one or more group-related entries on the ACL apply, DCE LFS grants the user all of the permissions accrued from the applicable group entries after filtering them through the **mask_obj**, if it exists. The user accrues permissions from all of the groups to which the user belongs.
4. The user matches the **other_obj** entry. If the **other_obj** entry applies, DCE LFS grants the user the permissions specified in the entry.
5. The user matches a **foreign_other** entry. If a **foreign_other** entry on the ACL applies, DCE LFS grants the user the specified permissions after filtering them through the **mask_obj**.
6. The user matches the **any_other** entry. If the **any_other** entry is on the ACL and it applies, DCE LFS grants the user the specified permissions after filtering them through the **mask_obj**.

7. The user matches no entry. If no entry applies, DCE LFS denies the user access.

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

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

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

```
user:dale:r-----
group_obj:rw----
```

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

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

14.1.2.1 ACL Evaluation for Local Access

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

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

14.1.3 Setting and Examining ACLs

The **acl_edit** command from the DCE Security Service is used to list and modify the ACLs of a DCE LFS object. The command can be used in interactive or command-line mode. This section documents some DFS-specific information about the command and provides a brief example of its use. (See the DCE Security Service portion of the *OSF DCE User's Guide and Reference* for complete details about using the command to set and examine an object's ACLs.)

In most respects, the operation of the **acl_edit** command with DCE LFS objects parallels its use with other types of DCE objects. To examine an ACL for a file or directory using the **list** subcommand, you must have the **x** (execute) permission on the directory in which the object resides, as well as on all directories that lead to that directory. To modify an entry on an ACL for a file or directory using **modify** or another subcommand, you must have the **c** (control) permission for the object, as well as the **x** (execute) permission on each directory that leads to the object.

Recall that only one entry of the same specificity can exist on an ACL; for example, only one **user** entry can exist for a given *username*. 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. If you want a user or group to retain the permissions already granted, you must include those permissions when you define the new entry for the user or group.

The following rules apply only to DCE LFS file and directory objects:

- The **user_obj**, **group_obj**, and **other_obj** entries must always exist.
- The **user_obj** entry must always explicitly retain the **c** permission.
- The **mask_obj** entry must exist if an entry other than **user_obj**, **group_obj**, or **other_obj** exists.

These rules restrict the use of the **acl_edit** command. Namely, if the **assign**, **delete**, **kill**, **modify**, or **substitute** subcommand of the **acl_edit** command is used to violate any of these restrictions, additional subcommands must be included to reinstate the necessary entries or permissions. Otherwise, you cannot save the modifications to the ACL.

Be especially careful when using the **acl_edit** command in interactive mode. The command allows you to violate these restrictions as long as you remain in interactive mode. However, it does not let you save the ACL with the **commit** or **exit** subcommand. Moreover, if you violate one of these restrictions in the process of making other valid changes, the command discards *all* changes when you use the **exit** subcommand in an attempt to save your changes and leave the **acl_edit** command. It is always better to use the **commit** subcommand to save changes to an ACL; if you violate a restriction, the command reports that it cannot save the ACL, but it does not discard your changes.

The remainder of this section describes the use of the **acl_edit** command to display and modify a directory's ACL. The **acl_edit** command's **list** subcommand is used to display an object's ACL. The following example shows the output of the **list** subcommand (invoked as **-l** in command-line mode) when it is used to display the ACL for the directory **drafts**:

```
$ acl_edit /.../abc.com/fs/doc/drafts -l

# SEC_ACL for /.../abc.com/fs/doc/drafts:
# Default cell = /.../abc.com
mask_obj:r-x---
```

```
user_obj:rwxcid
user:dale:rwx-id            #effective:r-x---
group_obj:rwx---            #effective:r-x---
group:writers:rwx---        #effective:r-x---
other_obj:rwx---
```

The first lines of the output display the pathname and default cell of the object. The remainder of the lines show the ACL entries for the object. If the **mask_obj** entry restricts the permissions granted by another entry, the permissions that remain after filtering through the mask are labeled `#effective`.

In this example, the permissions (**rwx-id**) granted to **dale** are restricted by the **mask_obj** to **r** and **x**. Users belonging to the group **writers** are, like **dale**, restricted to **r** and **x** access. The owner of the directory (**user_obj**) retains all of the specified permissions (**rwxcid**) because **user_obj** is not filtered by **mask_obj**.

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

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

The following example performs both of these operations with one invocation of the **acl_edit** command. It uses the **modify** subcommand (invoked as **-m** in command-line mode) to add an entry for **pierette** to the ACL. It also uses the **-c** option to direct the **acl_edit** command to recalculate the permissions granted by the **mask_obj** entry to include those to be granted to **pierette**. Alternatively, the **-n** option could be used to prevent the **acl_edit** command from recalculating the permissions granted by the **mask_obj** entry, but this would cause the **mask_obj** entry to restrict **pierette**'s permissions. The command fails unless one of the two options is included. (The **acl_edit** command dynamically recalculates the **mask_obj** entry as necessary when new entries are added to an ACL. By default, it refuses to readjust the **mask_obj** entry if doing so would grant

currently masked permissions to another entry. In such cases, you must include the **-c** or **-n** option to specify the command's actions with respect to the **mask_obj** entry.)

```
$ acl_edit /.../abc.com/fs/doc/drafts -c -m user:pierette:rwxid
```

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

```
$ acl_edit /.../abc.com/fs/doc/drafts -l

# SEC_ACL for /.../abc.com/fs/doc/drafts:
# Default cell = /.../abc.com
mask_obj:rwx-id
user_obj:rwxcid
user:dale:rwx-id
user:pierette:rwx-id
group_obj:rwx---
group:writers:rwx---
other_obj:rwx---
```

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

14.1.4 ACL Interaction with UNIX Mode Bits

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

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

Because DCE ACLs can be used only with objects in DCE LFS filesets, mode bits are the only form of protection associated with objects in most non-LFS filesets. In DCE LFS filesets, all file and directory objects can have both UNIX mode bits and DCE ACLs. Note that all objects always have UNIX mode bits, but they do not necessarily have ACLs. (See Section 14.1.5 for more details.)

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

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

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

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

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

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

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

```
mask_obj:r-----
user_obj:rwxc--
group_obj:r-x---        #effective:r-----
other_obj:rwx---
```

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

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

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

14.1.5 Initial Protection of a New File or Directory

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

Object ACL

> Controls access to the object itself. By default, this is the ACL that the **acl_edit** command displays or modifies when it is issued.

Initial Object Creation ACL

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

Initial Container Creation ACL

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

A directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL can exist independently of one another; they do not need to exist at all. A given directory can have all, some, or none of these ACLs.

The type of file system protection, ACLs or UNIX mode bits, initially used for a new file or directory object depends on whether the parent directory of the new object has the appropriate Initial Creation ACL, as follows:

- If a new object's parent directory has the appropriate Initial Creation ACL, the new object inherits an Object ACL as its form of protection. The new object also has mode bits, but the Object ACL supplements these bits. Recall that DCE LFS ensures that the object's mode bits and its ACL permissions are always synchronized.
- If a new object's parent directory does *not* have the appropriate Initial Creation ACL, the new object initially has no Object ACL; the object relies on mode bits as its only form of protection. If they are not inherited, ACLs can be explicitly created with the **acl_edit** command. (See Section 14.1.6 for information about using the **acl_edit** command to create a directory's initial ACLs.)

  Note: An Object ACL is always created for a file or directory that is created by a foreign user, even if the parent directory does not have the appropriate Initial Creation ACL. (See Sections 14.1.5.3 and 14.1.5.4 for more information.)

The following subsections describe how the initial protections of a new object are derived. The first subsection discusses an ACL's default cell, which plays an important role in determining ACL inheritance. The following subsections describe ACL inheritance for objects created by local users and objects created by foreign users; note that both of these subsections assume that the parent directory has the proper Initial Creation

ACL. The final subsection discusses how the UNIX mode bits are determined for an object whose parent directory does not have the appropriate Initial Creation ACL.

14.1.5.1 Default Cell of an ACL

To fully understand ACL inheritance, you need to understand the concept of an ACL's default cell. Each ACL has a default cell that names the cell with respect to which the ACL is defined. The ACL's entries are evaluated with respect to the default cell. The default cell is not necessarily the cell in which the ACL exists. For example, an object in cell **abc.com** can have an ACL whose default cell is **def.com**.

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

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

The **foreign_user**, **foreign_group**, **foreign_other**, and **any_other** entry types refer to users whose local, or home, cells are different from an ACL's default cell. A user's local cell is the cell in whose Registry Database the user's principal and account are defined. With respect to ACLs, a foreign user is one whose local cell is different from the default cell of the ACL, which may or may not be the cell in which the ACL resides.

To determine the default cell of an ACL, list the ACL with the **list** subcommand of the **acl_edit** command. The following line of the command's output clearly indicates the ACL's default cell:

```
# Default cell = /.../cell_name
```

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

```
# Default cell = /.../abc.com
```

When a file or directory object is initially created, the default cell of its Object ACL is set as the local cell of the user who creates the object. The default cells of a directory's Initial Object Creation ACL and Initial Container Creation ACL are also set as the default cell of the user who creates the object. The default cell of an object that does not have an Object ACL is the cell in which the object resides.

To change the default cell of an ACL, use the **cell** subcommand of the **acl_edit** command. You can make the default cells of a directory's Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL different from one another. However, changing each of a directory's ACLs to have different default cells can make it difficult to predict the effects of ACL inheritance and can lead to confusing results; therefore, it is not recommended. Also, because the default cell of an object's Object ACL is determined by the local cell of the user who creates the object, not by the default cell of the Initial Creation ACL that the object inherits, changing the default cell of an Initial Creation ACL is of limited utility.

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

Note that changing an ACL's default cell with the **acl_edit cell** command changes the meaning of the ACL's entries. For example, the **other_obj** entry no longer applies to users from the former default cell; it now applies to users from the new default cell. Also, entry types such as **user_obj** and **user**, which are defined with respect to an ACL's default cell, can now give permissions to different users in the new default cell.

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

14.1.5.2 ACL Inheritance for Objects Created by Local Users

When an object is created in a directory that has the appropriate Initial Creation ACL, DCE LFS uses the intersection of the following information to determine the Object ACL that it creates for the object:

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

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

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

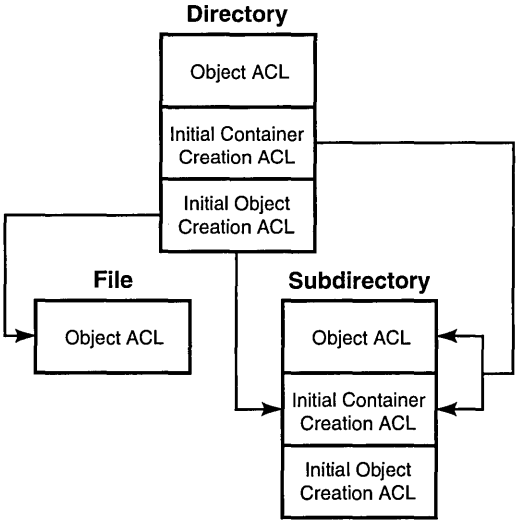
  If the **mask_obj** entry does not exist on the parent's Initial Object Creation ACL, the **r**, **w**, and **x** permissions for the file's **group_obj** entry are defined as the intersection of the **group** mode bits specified when the file is created and the corresponding permissions of the **group_obj**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

- The **r**, **w**, and **x** permissions for the file's **other_obj** entry consist of the intersection of the **other** mode bits specified when the file is created and the corresponding permissions of the **other_obj** entry of its parent directory's Initial Object Creation ACL. The **c**, **i**, and **d** permissions for the file's **other_obj** entry are copied directly from the **other_obj** entry of the parent's Initial Object Creation ACL.
- All other entries included on the parent directory's Initial Object Creation ACL are copied directly to the file's Object ACL.

DCE LFS uses the same algorithm to determine the initial entries and permissions for a subdirectory's Object ACL, but it uses the parent directory's Initial Container Creation ACL instead of its Initial Object Creation ACL. The subdirectory also inherits its parent's Initial Container Creation ACL as its Initial Container Creation ACL, and it inherits its parent's Initial Object Creation ACL as its Initial Object Creation ACL. The subdirectory inherits these Initial Creation ACLs unchanged from its parent directory. (See Section 14.1.5.4 for information about how the initial protections are determined for objects that do not inherit ACLs.)

Figure 14-1 illustrates ACL inheritance for files and directories.

Figure 14–1. ACL Inheritance



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

```
$ acl_edit /.:/fs/usr/rajesh -l

# SEC_ACL for /.:/fs/usr/rajesh:
# Default cell = /.../abc.com
mask_obj:rwx-id
user_obj:rwxcid
user:vijay:rwx-id
group_obj:r-x---
other_obj:r-x---
```

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

```
$ acl_edit /.:/fs/usr/rajesh -l -io

# Initial SEC_ACL for objects created under: /.:/fs/usr/rajesh:
# Default cell = /.../abc.com
mask_obj:rw----
user_obj:rw-c--
user:pierette:rw----
group_obj:r-----
other_obj:r-----
```

```
$ acl_edit /.:/fs/usr/rajesh -l -ic

# Initial SEC_ACL for directories created under: /.:/fs/usr/rajesh:
# Default cell = /.../abc.com
mask_obj:rwx-id
user_obj:rwxcid
user:pierette:rwx-id
group_obj:r-x---
other_obj:r-x---
```

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

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

```
$ acl_edit /.:/fs/usr/rajesh/myfiles -l

# SEC_ACL for /.:/fs/usr/rajesh/myfiles:
# Default cell = /.../abc.com
mask_obj:rwx-id
user_obj:rwxcid
user:pierette:rwx-id
group_obj:r-x---
other_obj:r-x---
```

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

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

14.1.5.3 ACL Inheritance for Objects Created by Foreign Users

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

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

When a user creates an object, ACL inheritance occurs as described in Section 14.1.5.2. However, if the user is a foreign user with respect to the appropriate Initial Creation ACL, entries inherited from the Initial Creation ACL are modified as follows:

- The **mask_obj** entry remains unchanged. It applies to the same entries on both the Initial Creation ACL and the new Object ACL.
- The **user_obj**, **group_obj**, and **other_obj** entries remain unchanged, but they are defined with respect to the default cell of the new Object ACL, not the default cell of the Initial Creation ACL. The **user_obj** entry specifies the permissions granted to the user who creates the object (the user whose local cell dictates the default cell of the ACL).
- Any **user** and **group** entries are changed to **foreign_user** and **foreign_group** entries because they are not defined with respect to the default cell of the new Object ACL.
- Any **foreign_user** and **foreign_group** entries that are defined with respect to the default cell of the new object ACL are changed to **user** and **group** entries.

- Any **foreign_user** and **foreign_group** entries that are defined with respect to neither the default cell of the Initial Creation ACL nor the default cell of the new Object ACL remain unchanged.
- Any **foreign_other** entries and the **any_other** entry remain unchanged.

If a user who is foreign with respect to the default cell of a directory's Object ACL creates an object in the directory, an Object ACL is created for the new object even if the parent directory does not have the appropriate Initial Creation ACL. In this case, the Object ACL must be created to record the fact that the new object's default cell is different from the cell in which the object resides. The permissions granted by the Object ACL are based on the UNIX mode bits specified at the system call level when the object is created and on the value of the UNIX **UMASK** variable of the creating process. (See Section 14.1.5.4 for more information.) Note that because an unauthenticated user is treated as a user from an unknown foreign cell, an Object ACL is always created for an object created by an unauthenticated user.

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

```
$ acl_edit /.:/fs/usr/srivas -l

# SEC_ACL for /.:/fs/usr/srivas:
# Default cell = /.../abc.com
mask_obj:rwx-id
user_obj:rwxcid
user:vijay:rwx-id
foreign_user:/.../def.com/andi:rwx-id
foreign_user:/.../ghi.com/pervaze:r-x---
group_obj:r-x---
other_obj:r-x---
foreign_other:/.../def.com:r-x---
```

The following **acl_edit** commands display the Initial Object Creation ACL and Initial Container Creation ACL of the directory:

```
$ acl_edit /.:/fs/usr/srivas -l -io

# Initial SEC_ACL for objects created under: /.:/fs/usr/srivas:
# Default cell = /.../abc.com
mask_obj:rw----
user_obj:rw-c--
user:pierette:rw----
foreign_user:/.../def.com/andi:rw----
foreign_user:/.../ghi.com/pervaze:r-----
group_obj:r-----
other_obj:r-----
foreign_other:/.../def.com:r-----

$ acl_edit /.:/fs/usr/srivas -l -ic

# Initial SEC_ACL for directories created under: /.:/fs/usr/srivas:
# Default cell = /.../abc.com
mask_obj:rwx-id
user_obj:rwxcid
user:pierette:rwx-id
foreign_user:/.../def.com/andi:rwx-id
foreign_user:/.../ghi.com/pervaze:r-x---
group_obj:r-x---
other_obj:r-x---
foreign_other:/.../def.com:r-x---
```

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

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

subdirectory inherits the following entries from the Initial Container Creation ACL of the parent directory:

```
$ acl_edit /.:/fs/usr/srivas/andi_files -l

# SEC_ACL for /.:/fs/usr/srivas/andi_files:
# Default cell = /.../def.com
mask_obj:rwx-id
user_obj:rwxcid
foreign_user:/.../abc.com/pierette:rwx-id
user:andi:rwx-id
foreign_user:/.../ghi.com/pervaze:r-x---
group_obj:r-x---
other_obj:r-x---
foreign_other:/.../def.com:r-x---
```

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

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

- The **foreign_other:/.../def.com** entry is unchanged; it continues to apply to users from the cell **def.com**. However, because the default cell of the ACL is now **def.com**, users from the cell who are not granted permissions from specific user or group entries are now granted permissions from the **other_obj** entry. The **foreign_other:/.../def.com** entry remains on the ACL, but as long as the default cell of the ACL is **def.com**, this **foreign_other** entry does not determine the permissions granted to users from the **def.com** cell.

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

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

```
$ acl_edit /.:/fs/usr/srivas/andi_files -l -io

# Initial SEC_ACL for objects created under: /.:/fs/usr/srivas/andi_files:
# Default cell = /.../def.com
mask_obj:rw----
user_obj:rw-c--
foreign_user:/.../abc.com/pierette:rw----
user:andi:rw----
foreign_user:/.../ghi.com/pervaze:r-----
group_obj:r-----
other_obj:r-----
foreign_other:/.../def.com:r-----
```

```
$ acl_edit /.:/fs/usr/srivas/andi_files -l -ic

# Initial SEC_ACL for directories created under: /.:/fs/usr/srivas/andi_files:
# Default cell = /.../def.com
mask_obj:rwx-id
user_obj:rwxcid
foreign_user:/.../abc.com/pierette:rwx-id
user:andi:rwx-id
foreign_user:/.../ghi.com/pervaze:r-x---
group_obj:r-x---
other_obj:r-x---
foreign_other:/.../def.com:r-x---
```

14.1.5.4 Mode Bits for New Objects That Do Not Inherit ACLs

The ACL inheritance described in Section 14.1.5.2 applies only if the directory in which a file or directory object is created has the appropriate Initial Creation ACL. If the parent directory of a new object does not have the appropriate Initial Creation ACL, the object does not inherit an Object ACL. It initially has no Object ACL and is protected only with UNIX mode bits.

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

- The UNIX mode bits specified at the system call level (with the UNIX **open**(), **creat**(), or **mkdir**() system call) when the object is created. For example, when the UNIX **touch** command is used to create an object, the resulting system call usually specifies the **user**, **group**, and **other** mode bits as **r** and **w**.
- The value of the UNIX **UMASK** variable of the process that creates the object. The **UMASK** variable filters the mode bits initially assigned to an object; the variable is defined as the octal complement of the allowable mode bits. For example, if a user creates an object, the value of the **UMASK** variable of the user's login process filters the mode bits assigned to the object.

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

Similarly, the system call to create a new directory usually grants **r**, **w**, and **x** access to **user**, **group**, and **other**; the **UMASK** variable commonly restricts **group** and **other** to only **r** and **x** access. When the directory is created, **user** has the **r**, **w**, and **x** bits, while **group** and **other** have just the **r** and **x** bits. Because its parent directory has no Initial Container Creation ACL, the directory is also created without an Initial Container Creation ACL. If the parent directory has an Initial Object Creation ACL, the new directory inherits it; otherwise, the new directory is created without an Initial Object Creation ACL.

Note: An explicit Object ACL is created for a file or directory that is created by a user whose local cell is different from the default cell of the parent directory's Initial Object Creation ACL or Initial Container Creation ACL. The permissions granted by the Object ACL are based on the UNIX mode bits specified at the system call level when the object is created and the value of the UNIX **UMASK** variable of the creating process. Because an unauthenticated user is treated as a user from an unknown foreign cell, an explicit Object ACL is always created for an object created by an unauthenticated user.

14.1.6 Initial ACLs of a New Fileset

The root directory of a newly created DCE LFS fileset has null ACLs. Until the **acl_edit** command is used to create the Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL for the root directory, the directory has no ACLs; it is protected only with UNIX mode bits. Files and subdirectories created in the directory are assigned mode bits according to the usual file system semantics described in Section 14.1.5.4.

A DCE LFS fileset can include many files and directories that never have ACLs. However, this approach fails to take advantage of the enhanced security available with DCE ACLs. Therefore, it is important to use the

acl_edit command to create the Object ACL, Initial Object Creation ACL, and Initial Container Creation ACL for the root directory of a fileset *before* other objects are created in the directory.

For the root directory of a new DCE LFS fileset, **user**, **group**, and **other** all receive the UNIX **r**, **w**, and **x** permissions. When the **acl_edit** command is initially used to view the Object ACL for a newly created root directory, DCE LFS uses these mode bits to construct the following implicit ACL for the directory:

```
user_obj:rwxcid
group_obj:rwx-id
other_obj:rwx-id
```

Note that the entries in the implicit ACL have the **i** and **d** permissions. This is true because DCE LFS expands the **w** mode bit of a directory's implicit ACL to grant the **w**, **i**, and **d** ACL permissions. Until a directory has an Object ACL, DCE LFS must perform this expansion to allow for the creation of objects in the directory; without the **i** and **d** ACL permissions, the directory would effectively be read-only. Once the ACL exists, DCE LFS maps the **w** mode bit to just the **w** ACL permission, as described in Section 14.1.4. Because the **i** and **d** ACL permissions are meaningless for a file, DCE LFS does not expand the **w** mode bit to grant the **i** and **d** permissions on the implicit ACL of a file.

When the **acl_edit** command is invoked with the **-io** or **-ic** option to first view the Initial Object Creation ACL or Initial Container Creation ACL for the directory, DCE LFS uses the value of the UNIX **UMASK** variable of the process that invokes the command to derive implicit entries and permissions for these ACLs. This is true of any directory that does not have these Initial Creation ACLs when the **acl_edit** command is used to view them.

For example, if the **UMASK** variable masks the **w** mode bit for the **group** and **other** users, the implicit Initial Container Creation ACL for the directory is as follows:

```
user_obj:rwxcid
group_obj:r-x---
other_obj:r-x---
```

In this case, the implicit Initial Object Creation ACL for the directory is as follows:

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

Note that in all cases the **user_obj** entry is explicitly granted the **c** permission, meaning that the owner of the directory can change the directory's ACLs. Cell administrators for the File Server machine on which the fileset resides (members of the group specified with the **-admingroup** option of the **fxd** command issued on the machine) can also change the ACLs of the directory.

The root directory's ACLs remain implicit until the **commit** or **exit** subcommand of the **acl_edit** command is used to save them, at which point explicit ACLs are created for the directory. The Object ACL created with the **acl_edit** command always matches the directory's mode bits. The Initial Object Creation ACL and Initial Container Creation ACL can vary, depending on the value of the **UMASK** variable of the process that saves these ACLs with the **acl_edit** command.

14.1.7 Suggested Initial ACLs for a New Fileset

Cell administrators need to create the ACLs for the root directory of a new DCE LFS fileset. They should also manipulate the root directory and its ACLs to assign the directory the proper owner and give its ACL entries the appropriate permissions. The owner of a fileset's root directory is initially set to **root**. A cell administrator must use the UNIX **chown** command or its equivalent to make the user who is to own the fileset the owner of the directory, thus granting that individual the **c** permission associated with the **user_obj** entry. A cell administrator should also use the UNIX **chgrp** command or its equivalent to change the owning group, as required.

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

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

```
$ acl_edit /.../abc.com/fs/usr/pierette -l

# SEC_ACL for /.../abc.com/fs/usr/pierette:
# Default cell = /.../abc.com
user_obj:rwxcid
group_obj:r-x---
other_obj:r-x---
```

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

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

14.2 Using Groups with DFS

Information about groups is maintained in the Registry Database by the DCE Security Service. (See the *OSF DCE Administration Guide—Core Components* for complete details about creating and maintaining groups.)

Using groups allows you to assign permissions to several users at one time, rather than assigning them individually. You can create user groups or special interest groups (for example, a group of all of the people from one department or a group of people who are working on one project) and then assign that group access to the appropriate files and directories.

You can also use groups to specify individuals who are permitted to perform administrative tasks; these individuals are specified in DFS administrative lists. DFS uses administrative lists to determine who is

authorized to issue commands that affect filesets and server processes. Through administrative lists, you can precisely control the security in the administrative domains in your cell. This chapter does not discuss the management of administrative lists in detail. (See Chapter 15 for details about creating and maintaining administrative lists.)

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

14.2.1 Creating and Maintaining Groups

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

The collection of groups to which a user belongs is called a project list. A user acquires the access permissions granted to each group on the user's project list. To assign a user to a group, use the **rgy_edit** command to add the user's principal name to the group's membership list in the Registry Database. (See the *OSF DCE Administration Guide—Core Components* for information about the **rgy_edit** command.)

14.2.2 Using Groups with ACLs, Administrative Lists, and Commands

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

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

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

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

14.2.3 Suggestions for Administrative Groups

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

Each type of server process is associated with an administrative list, which allows you to differentiate between users who perform different administrative tasks. For example, administrative users who start and stop server processes need to be included on different administrative lists from users who manipulate filesets. (See Chapter 15 for details about the administrative tasks associated with each administrative list.)

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

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

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

While inclusion in this administrative group is similar in many respects to being logged in as **root**, the two are not equivalent. A user who is logged into the local machine as **root** can perform different operations on a file or directory, depending on whether the user accesses the object via its DCE pathname or via its local pathname.

The first way a user can access a file or directory is via the object's DCE pathname. For DCE access, DFS treats a user who is logged into the local machine as **root** but not authenticated to DCE as an unauthenticated user; in this case, the **root** user has no special privileges. If the user is also authenticated to DCE as **root**, DFS treats the user according to **root**'s DCE identity. The DCE identity **root** effectively has **root** privileges for

data in all exported non-LFS filesets in the cell, which is a serious security risk. Use the DCE **root** identity very cautiously or disable it altogether.

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

Being a member of the **fxd** administrative group allows you to perform any operation on a file or directory in an exported fileset, but you may have to change the file's or directory's protections first. Being logged into the local machine as **root** lets you perform any operation on a file or directory in a locally mounted fileset immediately.

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

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

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

It could also include a third administrative group (**cell_file_system**). The members of this group would be a highly trusted subset of the members of the **cell_fileset** group. Table 14-3 lists the groups associated with each administrative list when only two groups are used and the groups associated with each administrative list when three groups are used. It also describes the function of the groups included in each list.

Table 14–3. Suggested Groups for Administering a Single-Domain Cell

Administrative List	With Two Groups	With Three Groups	Function
admin.bos	**cell_fileset**	**cell_file_system**	Manage server processes on each server machine
admin.fl	**cell_fileset**	**cell_fileset**	Create server and fileset entries in the Fileset Location Database on each Fileset Database machine
admin.ft	**cell_fileset** **cell_servers**	**cell_fileset** **cell_servers**	Manage filesets on each File Server machine; move filesets between File Server machines
admin.bak	**cell_fileset**	**cell_fileset**	Modify the Backup Database on each Backup Database machine
admin.up	**cell_servers**	**cell_servers**	Allow **upclient** processes to obtain files from **upserver** processes on server machines

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

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

An additional usage of the **-owner** option of the **fts crserverentry** command and the **-admingroup** option of the **fxd** command is to allow owners of local workstations to export data from their local disks to the global namespace. In this case, a group consisting of the owners of a local workstation is specified with these options when a server entry is created for the workstation and when the File Exporter is initialized on the machine. (See Chapter 17 for more information about creating server entries.)

Chapter 15

Using Administrative Lists and Keytab Files

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

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

You can use the **rgy_edit** command, which is provided as part of the DCE Security Service, to create administrative groups. You can then use the **bos addadmin** command to place the groups on administrative lists. (See Chapter 14 for more information about groups.)

Each DFS server machine also has a keytab file. The file contains server encryption keys, at least one of which is also stored in the cell's Registry Database. Keytab files are used to provide security between server machines

and client machines. A server machine uses an encryption key from the keytab file to prove that it is a valid server to clients accessing data from it, as well as to other server machines from which it accesses data.

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

15.1 Standard Options and Arguments

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

- The **-server** *machine* option is the server machine on which the command is to execute. This option names the machine on which the administrative list or keytab file to be affected is stored. The BOS Server on this machine executes the command. Use the DCE pathname (for example, **/.../abc.com/hosts/fs1**) with all commands. This option can be used to execute a **bos** command on a server machine in a foreign cell.

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

- The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See Section 15.2.3 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.

- The **-localauth** option directs the **bos** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, **/.../abc.com/hosts/fs1/dfs-server**. Do not confuse a machine's DFS server principal with its unique **self** identity. (See Chapter 17 for information about DFS server principals.)

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

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

15.2 Using Administrative Lists

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

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

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

15.2.1 Administrative Lists

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

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

- The **admin.bak** list is associated with the Backup Server. It designates the users and groups allowed to issue commands in the **bak** command suite. These commands are used to configure the Backup System and to dump and restore data. The Backup Database, like the FLDB, is typically replicated to several different machines in the cell. Therefore, you need to ensure that the **admin.bak** lists on all machines that house the Backup Database are identical.

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

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

15.2.2 Maintaining Administrative Lists

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

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

administrative list if the list does not already exist when the process starts, the **admin.bos** list *must* already exist when you issue the **bos addadmin** command.

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

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

To guarantee that all users and groups have the same privileges on all server machines, the same users and groups must be on the administrative lists that grant those privileges on each machine. If the same copy of an administrative list is not distributed to all machines in the domain, users can be prohibited from issuing commands on specific machines. For instance, suppose a user is listed in the **admin.ft** file on machine **fs1** but is not listed in the **admin.ft** file on machine **fs2**. The user can issue commands that affect filesets on **fs1**, but the user cannot issue commands that affect filesets on **fs2**.

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

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

You can remove an administrative list that you no longer need by including the **-removelist** option with the **bos rmadmin** command. If you use the command to remove the last member from an administrative list or if a list

contains no members when you issue the command, the **-removelist** option specifies that the list is to be removed. The option has no effect if the list is not empty.

15.2.2.1 Listing Principals and Groups in Administrative Lists

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

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

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

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

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

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

15.2.2.2 Adding Principals and Groups to Administrative Lists

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

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the administrative list to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list.

2. Issue the **bos addadmin** command to add principals, groups, or both to an administrative list:

   $ **bos addadmin -server** *machine* **-adminlist** *filename* [**-principal** *name...*] [**-group** *name...*] [**-createlist**]

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

   The **-principal** *name* option specifies the principal name of each user or server machine to be added to the list. A user from the local cell can be specified by a full or abbreviated principal name (for example, */.../cellname/username* or just *username*); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, */.../cellname*/**hosts**/*hostname*/**self** or just **hosts**/*hostname*/**self**); a server machine from a foreign cell can be specified only by a full principal name.

   The **-group** *name* option specifies the name of each group to be added to the list. A group from the local cell can be specified by a full or abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

   The **-createlist** option specifies that the administrative list indicated with **-adminlist** is to be created if it does not already exist. Any principals or groups specified with the command are added to the new file; if no principals or groups are specified, the command creates an empty file. This option has no effect if the specified file already exists.

15.2.2.3 Removing Principals and Groups from Administrative Lists

To rename principals and groups for an administrative list, do the following:

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

   $ **bos rmadmin -server** *machine* **-adminlist** *filename* [**-principal** *name...*] [**-group** *name...*] [**-removelist**]

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

   The **-principal** *name* option specifies the principal name of each user or server machine to be removed from the list. A user from the local cell can be specified by a full or abbreviated principal name (for example, */.../cellname/username* or just *username*); a user from a foreign cell can be specified only by a full principal name. A server machine from the local cell can be specified by a full or abbreviated principal name (for example, */.../cellname/***hosts/***hostname/***self** or just **hosts/***hostname/***self**); a server machine from a foreign cell can be specified only by a full principal name.

   The **-group** *name* option specifies the name of each group to be removed from the list. A group from the local cell can be specified by a full or abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

   The **-removelist** option specifies that the administrative list indicated with **-adminlist** is to be removed if it is empty either when the

command is issued or after any principals or groups specified with the command are removed. This option has no effect if the specified file is not empty when the command is issued or after any indicated principals or groups are removed.

15.2.3 Disabling DFS Authorization Checking on a Server Machine

"DFS authorization checking" involves a server process checking the proper administrative list to ensure that the issuer of a command has the necessary administrative privilege to execute the command. If the issuer is a member of the list, the process performs the requested operation; if the issuer is not a member of the list, the process does not perform the operation.

By default, DFS authorization checking is enabled on every server machine. You can disable it on a machine by

- Including the **-noauth** option with the **bosserver** command when the BOS Server is started on the machine.
- Issuing the **bos setauth** command and specifying the machine with the command's **-server** option.
- Manually creating the zero-length file *dcelocal*/**var/dfs/NoAuth** on the local disk of the machine; the first two methods create this file automatically.

All DFS server processes, including the BOS Server, check for the presence of the **NoAuth** file when they are requested to perform an operation. They do not check for the necessary administrative privilege for a requested operation when the file is present. Consider disabling authorization checking on a machine in the following situations:

- During initial DFS installation, by including the **-noauth** option with the **bosserver** command. Before administrative lists have been created or users have been added to the lists, no one has the necessary privilege to issue an administrative command.
- If some component of the Security Service is unavailable, by manually creating the **NoAuth** file. If the **secd** process or a related security process is unavailable, the issuer of a command cannot acquire the security credentials necessary to allow DFS server processes to verify administrative privilege. In this case, the **-noauth** option must be

included with a command to bypass the unavailable Security Service. (See Section 15.2.3.1.)

- During server encryption key emergencies, by manually creating the **NoAuth** file. Improper keys may make it impossible for DFS server processes to verify a user's administrative privilege. The **-noauth** option can again be used to circumvent the security problems.
- To view the actual keys stored in a keytab file, by issuing the **bos setauth** command. If authorization checking is enabled, checksums are displayed rather than the actual keys. (See Section 15.3.)

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

15.2.3.1 Using the -noauth Option

Most DFS commands from the **bos** and **fts** suites have an optional **-noauth** option. Omitting the **-noauth** option from a command requires that authentication information be available about the issuer of the command; because it is optional, the option is always omitted by default. Including the **-noauth** option with a command directs the **bos** or **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. Include the **-noauth** option with a command if

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

You may want to include the **-noauth** option with a command that does not require administrative privilege if the command is to be issued as a **bos cron** process. (See Chapter 16 for more information about **bos** processes.)

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

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

15.2.3.2 Disabling or Enabling DFS Authorization Checking

To disable or enable DFS authorization checking, do the following:

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

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

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

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

15.3 Using Keytab Files

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

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

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

15.3.1 Maintaining Keytab Files

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

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

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

A keytab file must already exist before either of these commands can be used to add a key to it; keytab files are created with the **rgy_edit** command.

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

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine. Because reusing a version number currently in use in a keytab file can cause authentication failures between the processes on a server machine and clients communicating with them, an error is returned if you attempt to do so.
- + or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database. However, it may not be unique for the indicated principal in the keytab file on the specified machine, in which case it replaces the key currently associated with the principal/version number pair in the keytab file.

It is best to keep the key version numbers in sequence by choosing a number that is one greater than the current version number for the principal. Use the **bos lskeys** command to examine the key version numbers associated with the keys in a keytab file.

The **bos lskeys** command also displays a **checksum** with each key version number. A checksum is a decimal number derived by encrypting a constant with a key. Because displaying the checksum is adequate for most purposes (for example, when checking key version numbers presently in use), and because its display is less of a security risk, it is displayed rather than the actual key associated with a version number. Note that the actual keys can be viewed by first issuing the **bos setauth** command to disable DFS authorization checking on the server machine; however, because disabling DFS authorization checking creates a compromised state of security, it is not recommended.

After a new key has been added to a keytab file, the old key can be removed from the file. The **bos rmkey** command can be used to remove one or more keys from a keytab file. Removing the key currently in use in the Registry Database or any other key still being used for client/server communications can cause authentication failures between server processes and clients. Tickets based on a removed key are invalidated; new tickets based on a new key must be obtained to reestablish communications with the server process.

To prevent authentication failures, wait until all old tickets held by client machines expire before removing the old key. For example, if tickets held by clients expire after 2 hours, wait at least that long from the time the new key is added to remove the old key. If you are unsure of whether a key is still in use, use the **bos gckeys** command to delete, or "garbage collect," those keys from a keytab file that are no longer in use (obsolete).

Note: The BOS Server uses authenticated RPC for communications with clients. By default, it uses the packet privacy protection level with the **bos** key commands described in this chapter. However, this protection level is not available to everyone who uses DCE. If it is not available to you, the BOS Server uses the next-highest protection level, packet integrity. It displays the following message, reporting that it must use the packet integrity protection level because packet privacy is not available:

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

15.3.1.1 Listing Keys in Keytab Files

To list the keys in a keytab file, do the following:

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

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

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

15.3.1.2 Adding Keys to Keytab Files

To add a key to a keytab file, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be include in the **admin.bos** list on the machine on which the keytab file to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.
2. Verify that the DFS server principal of the machine whose keytab file is to be affected has the necessary permissions to alter entries in the Registry Database. (See the *OSF DCE Administration Guide—Core Components* for more information.)
3. Choose a key version number for the new key. If necessary, issue the **bos lskeys** command to check the version numbers of the keys in the appropriate machine's keytab file:

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

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

4. Create a new key in the keytab file with either the **bos genkey** command or the **bos addkey** command. The **bos genkey** command is the more secure of the two commands. It generates a random, octal string for use as the key. It also automatically updates the Registry Database in addition to adding the key to the keytab file.

   $ **bos genkey -server** *machine* **-kvno** *+_or_version_number* [**-principal** *name*]

   The **-kvno** *+_or_version_number* option is the key version number of the new key. Valid arguments for this option are

   - An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine.
   - + or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database, but it may not be unique for the indicated principal in the keytab file on the specified machine.

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

   The **bos addkey** command is less secure because it requires you to enter a string to be converted into the key. However, you can include the **-localonly** option with the command to add the key to the keytab file without updating the Registry Database, which is useful for certain server encryption key emergencies.

   $ **bos addkey -server** *machine* **-kvno** *+_or_version_number* **-password** *string* [**-principal** *name*] [**-localonly**]

The **-kvno** *+_or_version_number* option is the key version number of the new key. Valid arguments for this option are

- An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine.
- + or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database, but it may not be unique for the indicated principal in the keytab file on the specified machine.

The **-password** *string* option is a character string that is to be converted into an octal string. The string can include any characters, including spaces if it is enclosed in " " (double quotes).

The **-principal** *name* option is the principal name with which the new key is to be associated. The default is the DFS principal name of the machine specified with **-server**.

The **-localonly** option specifies that the key is to be added to the keytab file on the machine indicated by **-server**, but the Registry Database is not to be updated.

5. If you added the key to the keytab file using the **bos addkey** command and its **-localonly** option, use the **rgy_edit** command to add the key to the Registry Database when necessary.

15.3.1.3 Removing Specific Keys from Keytab Files

To remove a specific key from a keytab file, do the following:

1. Verify that you have the necessary privilege to issue the command. You must be included in the **admin.bos** list on the machine on which the keytab file to be affected is located. If necessary, issue the **bos lsadmin** command to check the **admin.bos** list on the appropriate machine.

2. Remove one or more keys from the keytab file with the **bos rmkey** command:

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

   The **-kvno** *version_number* option is the key version number of each key to be removed for the indicated principal. Valid arguments for this option are integers in the range 1 to 255.

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

15.3.1.4 Removing All Obsolete Keys from Keytab Files

To remove all obsolete keys from a keytab file, do the following:

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

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

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

15.3.2 Handling Server Encryption Key Emergencies

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

A second type of server encryption key emergency can result from the current key becoming corrupted. In this case, server processes using the key cannot decrypt the information used in client/server communications, bringing all activity involving those processes to a halt. You must remove the corrupted key from the keytab file, but you do not need to reboot the server machine. From a security perspective, this type of emergency is less severe than one resulting from a compromised key, but it requires immediate attention nonetheless.

To resolve encryption key emergencies, you must add a new server key to both the keytab file on the machine and the Registry Database. You must turn off DFS authorization checking when handling key emergencies. Because disabling DFS authorization checking is a severe security risk, disable authorization checking for a minimal amount of time.

The emergency procedure requires you to be logged into the affected server machine as **root** to create the **NoAuth** file and to reboot the machine. Many of the steps in the following procedure were detailed in previous sections of this chapter. (See Chapter 16 for a description of the **bos shutdown** command.)

Note: Rebooting is not necessary when replacing a corrupted key. It may not always be necessary when dealing with a compromised key; for example, it may be sufficient simply to restart any processes associated with the compromised key. However, rebooting the machine is the safest way to terminate all unauthorized communications.

1. Log in as **root** on the affected machine.
2. Disable DFS authorization checking by creating the *dcelocal*/**var/dfs/NoAuth** file. It is usually recommended that you use the **bos setauth** command to create the **NoAuth** file. However, because the server encryption key emergency can make it impossible

to issue **bos** commands, create the file with the **touch** command (or its equivalent).

3. Use the **bos lskeys** command to check the key version numbers currently in use, using the **-noauth** option to employ an unprivileged identity as the identity of the issuer of the command:

   # **bos lskeys -server** *machine* [**-principal** *name*] **-noauth**

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

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

4. Create the new key with the **bos genkey** command, specifying a new key version number for the key with the **-kvno** option and again using the **-noauth** option:

   # **bos genkey -server** *machine* **-kvno** +_*or*_*version*_*number*
   [**-principal** *name*] **-noauth**

   The **-kvno** +_*or*_*version*_*number* option is the key version number of the new key. Valid arguments for this option are

   - An integer in the range 1 to 255. The command uses the specified integer as the version number of the new key. The integer must be unique for the indicated principal in the keytab file on the specified machine.
   - + or **0** (zero). The command chooses an integer to serve as the version number of the new key. The integer it chooses is unique for the indicated principal in the Registry Database, but it may not be unique for the indicated principal in the keytab file on the specified machine.

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

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

5. Use the **bos rmkey** command to remove any old keys that are compromised. Specify the version number of each key to be removed with the **-kvno** option, and again use the **-noauth** option.

   # bos rmkey -server *machine* **-kvno** *version_number*... [**-principal** *name*] **-noauth**

   The **-kvno** *version_number* option is the key version number of each key to be removed for the indicated principal. Valid arguments for this option are integers in the range 1 to 255.

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

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

6. *If the emergency resulted from a compromised key*, issue the **bos shutdown** command to prepare to reboot the machine. You must reboot the machine to terminate all existing communications that are based on the compromised encryption key. The **bos shutdown** command directs the BOS Server to shut down the other DFS server processes running on the machine. Include the **-wait** option with the command to be sure that all processes have stopped before continuing.

   # bos shutdown -server *machine* **-wait -noauth**

   The **-wait** option delays the command shell prompt's return until the processes are stopped. If the option is omitted, the prompt returns immediately, even if the processes are not yet stopped.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

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

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

   The **-authchecking on** option enables DFS authorization checking by removing the **NoAuth** file from the machine specified with

-server; **-authchecking off** disables authorization checking by creating the **NoAuth** file on the machine specified with **-server**.

The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

8. *If the emergency resulted from a compromised key*, issue the appropriate reboot command (**/etc/reboot** or its equivalent) for the machine to be rebooted. For example:

```
# /etc/reboot
```

15.3.3 The rgy_edit Command and Keytab Files

The **rgy_edit** command provided by the DCE Security Service can also be used to manipulate encryption keys in keytab files. (See the *OSF DCE Administration Guide—Core Components* for more information about the **rgy_edit** command.) The following subcommands of the **rgy_edit** command are used to manage keytab files:

ktadd	Creates keys in a keytab file and, optionally, in the Registry Database
ktlist	Lists the keys in a keytab file
ktdelete	Removes keys from a keytab file

These commands perform functions similar to those of their counterparts in the **bos** command suite. However, they require that you be logged into the server machine whose keytab file you want to manipulate. Because the analogous **bos** commands require only that you be included in the **admin.bos** list on the machine whose keytab file you wish to manage, they allow for remote system administration.

Chapter 16

Monitoring and Controlling Server Processes

To provide efficient and correct operation, the processes that are running on DFS server machines in a cell must be configured properly. The Basic OverSeer Server (BOS Server) continually monitors and, if necessary, restarts the other server processes on a machine; you specify the processes that the BOS Server is to monitor. The BOS Server runs on all DFS server machines.

You also control server process status by issuing **bos** commands to perform routine maintenance or to correct errors the BOS Server cannot correct by itself. This chapter explains how to define a server machine's processes and how to start and stop them. The BOS Server can monitor and control processes other than DFS processes. However, the information in this chapter refers specifically to DFS server processes.

Do not use the BOS Server to control the following processes on a machine: **fxd**, **dfsd**, **dfsbind**, or **dfsexport**. The first two processes spawn kernel threads that, if continually restarted, can eventually result in system failure on the machine. The last two processes are usually executed only when a machine initially starts, and they must be started in the proper sequence with respect to other processes. It is recommended that all four of these processes be started by including a line in the proper initialization file (**/etc/rc** or its equivalent).

16.1 Process Entries in the BosConfig File

You define which processes the BOS Server monitors by creating process entries in the *dcelocal*/**var/dfs/BosConfig** file on the local disk of each server machine. The information in a process entry defines how the process is to run. You control the process status (**Run** or **NotRun**) by changing the entry with **bos** commands. When the BOS Server starts, it creates a **BosConfig** file with no process entries if the file does not already exist.

The order in which process entries are added to or appear in the **BosConfig** file is irrelevant. The BOS Server restarts multiple processes virtually simultaneously. However, do not depend on one process starting before another simply because its entry precedes that of the other process in the **BosConfig** file. The BOS Server has no control over how long a process takes to start.

Caution: Do not directly edit the information in the **BosConfig** file; use only the commands described in this chapter to alter the file. Directly editing the **BosConfig** file can result in changes to process entries of which the BOS Server is unaware. Such changes do not take effect until the BOS Server is restarted and again reads the file.

Each process entry includes the following information about its process:

- Its name. The name that appears in the **BosConfig** file for a process is the name used to refer to that process with any **bos** commands that require a process name.
- Its type. The type can be one of the following:
  - **simple**—A **simple** process is a continuous process that runs independently of any other processes on a server machine. All standard DFS processes are **simple** processes. This process has a single parameter: the command to be executed.
  - **cron**—A **cron** process, like a **simple** process, runs independently of other processes; however, a **cron** process runs periodically, not continuously. This process has two parameters: the first is the command that is to be executed; the second is the time that the command is to be executed.

- Its status flag. Status flags are for internal use only and do not appear in any output. The flag can have one of the following values:
  - — **Run**, meaning the process needs to run whenever possible. The BOS Server starts the process initially at reboot and restarts it automatically if it fails at any time. This flag is used to keep a process running at all times; for example, to ensure that the **ftserver** process on a File Server machine runs continuously. (The **Run** status flag appears in the **BosConfig** file as a 1.)
  - — **NotRun**, meaning the process never runs. The BOS Server never starts or automatically restarts the process; the process runs only when you instruct the BOS Server to start it. This flag is used to stop a process for an extended period of time; for example, to stop the **upclient** process from accessing new binary files while you test the current binaries. (The **NotRun** status flag appears in the **BosConfig** file as a 0.)
- Its command parameters. These parameters are used by the BOS Server to run the process.

The following output from the **bos status** command displays an entry from the **BosConfig** file. (See Section 16.4 for more information about the **bos status** command, which is used to list entries from the **BosConfig** file.)

```
Instance ftserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:02 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/ftserver'
```

It is possible for the BOS Server's memory state to change independently of the **BosConfig** file. The BOS Server checks the **BosConfig** file whenever it starts or restarts (in response to the **bos restart** command, at the general restart time, or at system reboot). At that time, the BOS Server transfers information from the file into memory and does not read the file again until it restarts.

Therefore, it is possible to use the **bos shutdown** command to stop a running process, even though its status flag in the **BosConfig** file is **Run**. Similarly, you can use the **bos startup** command to start a process running by setting its memory state status flag to **Run** without setting its status flag in the file to **Run**. The commands discussed in this chapter can affect the BOS Server's memory state, the information in the **BosConfig** file, or both.

Starting or stopping certain processes, either temporarily or permanently, has an effect on the other processes that run on the other server machines in your cell. For example, an **upserver** process must run on each System Control machine and Binary Distribution machine. If you start or stop the process on one machine, you must start it on a replacement System Control or Binary Distribution machine. You must also modify the **upclient** processes on the appropriate server machines so that they reference the new System Control or Binary Distribution machine.

16.2 Standard Information in this Chapter

The following subsections present options and arguments common to many of the commands described in this chapter. It also presents some common operations that are explained in the chapter or that can be useful when performing other operations.

16.2.1 Standard Options and Arguments

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

- The **-server** *machine* option is the server machine on which the command is to execute. This option names the machine on which the process to be affected is running. The BOS Server on this machine executes the command. Use the DCE pathname (for example, **/.../abc.com/hosts/fs1**) with all commands. This option can be used to execute a **bos** command on a server machine in a foreign cell.
- The **-process** *server_process* option is the process to be created, started, or stopped. The following names are recommended for DFS server processes, but a process can be given any name:

  — **ftserver**: The Fileset Server process

— **flserver**: The Fileset Location Server process

— **upclient**: The client portion of the Update Server that transfers binary files (such as those for server processes) from *dcelocal*/**bin** and transfers configuration files (such as administrative lists) from *dcelocal*/**var/dfs** on the System Control machine

— **upserver**: The server portion of the Update Server

— **repserver**: The Replication Server process

— **bakserver**: The Backup Server process

- The **-cmd** *cmd_line* option specifies the commands and parameters that the BOS Server uses. For a **simple** process, only one command line specifying the binary file's complete pathname is necessary. This can be the pathname of a DFS command or any other command to be executed. For example, the command "*dcelocal*/**bin/fts clonesys**" backs up every fileset in the file system. As this example shows, you must enclose the parameter in double quotes if it contains spaces, and you must specify the complete pathname for the command.

  For a **cron** process, *cmd_line* specifies the following two command parameters:

  — The *first parameter* is the command that the BOS Server executes. As with the sole parameter for a **simple** process, this parameter can be the complete pathname of the binary file for a DFS command or any other command to be executed. As the example for the **simple** process shows, you must enclose the parameter in double quotes if it contains spaces, and you must specify the complete pathname for the command.

  — The *second parameter* specifies the time at which the BOS Server is to execute the command. This parameter must also be surrounded with double quotes if it contains spaces. Valid values are

    — **never**. The command does not execute, but the process entry remains in the **BosConfig** file.

    — **now**. The command executes immediately, but it never executes again; the process entry is removed from the **BosConfig** file after the command is executed.

    — A specific day of the week at a specific time ("*day hh:mm*"). The command executes weekly at the specified day and time.

— A specific time (*hh:mm*). The command executes daily at the specified time.

If you specify a day, it must appear first, in lowercase letters. You can enter either the entire name or just the first three letters; for example, **sunday** or **sun**. When indicating a time, separate hours from minutes with a colon. You can use 24-hour time or 1:00 through 12:00 with **am** or **pm**; for example, **14:30** or **"2:30 pm"**. You must enclose the entry in double quotes if it contains spaces; for example, **"sun 2:30 pm"**.

- The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See Chapter 15 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.
- The **-localauth** option directs the **bos** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server**; for example, **/.../abc.com/hosts/fs1/dfs-server**. Do not confuse a machine's DFS server principal with its unique **self** identity. (See Chapter 17 for information about DFS server principals.)

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

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

16.2.2 Standard Commands and Operations

Some of the following commands and operations are described in many places in this chapter; others can prove useful when the operations in this chapter are performed. If a command or operation is described in detail here, it is not described in depth in later sections of this chapter where it is used.

16.2.2.1 Determining Administrative Privilege

To perform the majority of **bos commands**, the issuer must be listed in the **admin.bos** file on the machine used in the command. To determine the members of a list, issue the **bos lsadmin** command:

```
$ bos lsadmin -server machine -adminlist admin.bos
```

The **-adminlist admin.bos** option specifies that members of the **admin.bos** file are to be listed.

16.2.2.2 Examining Log Files

The **bosserver** process and most of the server processes it monitors generate log files. The log files record execution messages and error messages generated by the server processes as they execute. By default, the processes write the files to the *dcelocal*/**var/dfs/adm** directory, although some server processes can be instructed to write their log files to a different directory. A list of the log files and the processes that write them follows:

- The **BakLog** file is generated by the Backup Server process on each Backup Database machine.
- The **BosLog** file is generated by the BOS Server process on each server machine.
- The **FlLog** file is generated by the Fileset Location Server process on each Fileset Database machine.
- The **FtLog** file is generated by the Fileset Server process on each File Server machine.
- The **RepLog** file is generated by the Replication Server process on each server machine.
- The **UpLog** file is generated by the **upserver** process on each server machine that is running the server portion of the Update Server.

The **bos getlog** command can be used to examine any of these log files, including the **.old** versions created by the associated server processes. By default, the command looks in the *dcelocal*/**var/dfs/adm** directory for the log file that it is to display. It is not necessary to specify the full pathname

of a log file if it resides in the default directory. However, if the file resides elsewhere, the full pathname of the log file must be provided.

In addition, no privilege is necessary to view a log file that resides in the default directory. If the file resides in a different directory, the issuer of the command must be listed in the **admin.bos** file on the machine on which the file is located, which is specified by the **-server** option.

$ **bos getlog -server** *machine* **-file** *log_file*

The **-file** *log_file* option specifies the log file that is to be displayed. A simple filename is sufficient for a log file that resides in the *dcelocal*/**var/dfs/adm** directory. A full pathname is required for a log file that resides in a different directory.

16.3 Creating and Starting Processes

To start a new process on a server machine, use the **bos create** command to alter the **BosConfig** file. This adds a process entry for the new process to the **BosConfig** file and sets the status flag for the process to **Run** in both the file and the BOS Server's memory, making the effect immediate. You can use the command to create both **simple** and **cron** processes.

The server process name included in this command is used by the BOS Server to reference the process. It is also used in any subsequent **bos** commands that require a process name. The BOS Server adds it to the **BosConfig** file when it creates the process's entry. The name does not appear in process listings generated with the **ps** command or its equivalent.

16.3.1 Creating and Starting a simple Process

To create and start a **simple** process, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the process is to be started. If necessary, issue the **bos lsadmin** command to check.

2. Create an entry for the **simple** process in the **BosConfig** file, and start it:

   $ **bos create -server** *machine* **-process** *server_process* **-type simple -cmd** *cmd_line...*

   The **-type simple** option specifies this as a **simple** process.

   Following is an example **simple** process entry named **flserver** on the machine named **fs1**:

   $ **bos create /.../abc.com/hosts/fs1 flserver simple** *dcelocal*/**bin/flserver**

16.3.2 Creating and Starting a cron Process

To create and start a **cron** process, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the process is to be started. If necessary, issue the **bos lsadmin** command to check.
2. Create an entry for the **cron** process in the **BosConfig** file, and start it:

   $ **bos create -server** *machine* **-process** *server_process* **-type cron -cmd** *cmd_line...*

   The **-type cron** option specifies this as a **cron** process.

   Following is a sample **cron** process entry named **backup** on the machine named **fs1**. The **-localauth** option allows the unauthenticated process to use the DFS server principal of **fs1** to execute the privileged **fts clonesys** command.

   $ **bos create /.../abc.com/hosts/fs1 backup cron "***dcelocal*/**bin/fts clonesys -s /.../abc.com/hosts/fs1 -localauth" 5:30**

16.4 Listing Status and Machine Information

Use the **bos status** command to check the processes that are running on a server machine. The command causes the BOS Server to probe and determine the status of each process on the machine. It then displays output about the status of each process. It also displays appropriate messages if DFS authorization checking is disabled on the machine or if the machine's *dcelocal* directory or its contents are not protected appropriately.

The process information provided by the **bos status** command enables you to determine the role of the server machine (File Server machine, System Control machine, Binary Distribution machine, Fileset Database machine, Backup Database machine, or multiple machine roles). When you are using the **bos status** command to determine server machine roles, include the **-long** option to provide more detailed output.

16.4.1 Checking the Statuses of Processes on a Server Machine

Enter the **bos status** command to check the statuses of the processes on a server machine. Use the **-process** option to display the statuses of specific processes on the specified server machine, or omit the option to display the statuses of all processes on the machine.

$ **bos status -server** *machine* [**-process** *server_process...*] [**-long**]

The **-long** option indicates that more detailed information about the specified processes is to be displayed.

The command first displays the following line if DFS authorization checking is disabled on the machine (it does not display the line if DFS authorization checking is enabled):

```
Bosserver reports machine is not checking authorization.
```

It then displays the following line if the BOS Server finds that the *dcelocal* directory or a directory or file beneath it on the machine has protections that it believes are inappropriate:

```
Bosserver reports inappropriate access on server directories.
```

The BOS Server displays the message if the UNIX mode bits on the *dcelocal* directory and its contents do not enforce certain protections. The message usually indicates that users who should not be able to write to the *dcelocal* directory and its subdirectories have write access. The BOS Server displays the message as a courtesy to the user; it does nothing to change the protections, nor does it fail if the protections are violated. (See the description of the **bos status** command in the DCE DFS portion of the *OSF DCE Administration Reference* for information about the protections the BOS Server wants to see enforced.)

The command then displays status information about the processes on the machine. The possible statuses for any process include

- `currently running normally`—For a **simple** process, this means it is currently running; for a **cron** process, this means it is scheduled to run.
- `temporarily enabled`—The status flag for the process in the **BosConfig** file is **NotRun**, but the process has been enabled with the **bos startup** or **bos restart** command.
- `temporarily disabled`—Either the **bos shutdown** command was used to stop the process, or the BOS Server quit trying to restart the process, in which case the message `stopped for too many errors` also appears.
- `disabled`—The status flag for the process in the **BosConfig** file is **NotRun**, and the process has not been enabled.
- `has core file`—The process failed or produced a core file at some time. This message can appear with any of the other messages. Core files are stored in *dcelocal*/**var/dfs/adm**. The name of the core file indicates the process that failed; for example, **core.ftserver**.

The output for a **cron** process includes an auxiliary status message, reporting when the command is next scheduled to execute.

The following additional information is displayed when the **-long** option is used:

- The process type (**simple** or **cron**).
- How many **proc starts** occurred; **proc starts** occur when the process is started or restarted by the current BOS Server.
- The time of the last **proc start**.

- The exit time and error exit time when the process last failed. This appears only if the process failed while the BOS Server was running. (Provided the BOS Server was running both when the process was started and when it failed, the BOS Server can provide this information for any process that has an entry in the **BosConfig** file.)
- The command and its options that are used by the BOS Server to start the process.

The following examples show two executions of the **bos status** command on the same server machine. The first example shows the output displayed when the **-long** option is omitted from the command.

```
$ bos status /.../abc.com/hosts/fs4

Instance ftserver, currently running normally.
```

The second example shows the output displayed when the **-long** option is included with the command.

```
$ bos status /.../abc.com/hosts/fs4 -long

Instance ftserver, (type is simple) currently running normally.
Process last started at Fri Nov 22 05:36:02 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/ftserver'
```

16.4.2 Determining Server Machine Roles

The following instructions can help you use the **bos status** command to determine which server machines are filling the various machine roles in your cell or domain. The instructions assume that your cell is configured according to the installation and configuration instructions for your system; for example, they assume that all machines, except the System Control machine, are running a client portion of the Update Server that references the *dcelocal*/**var/dfs** directory on the System Control machine. If your server machines are not configured in this manner, these instructions may not help you to determine the roles of the machines.

To determine whether a server machine is a System Control machine, a Binary Distribution machine, or neither of the two types of machines, issue

the **bos status** command on the machine with **upserver** as the argument for the **-process** option. The output from the command indicates only whether the machine is a System Control machine, a Binary Distribution machine, or neither of the two; a machine that fits neither of the two roles can be a File Server machine, a Fileset Database machine, a Backup Database machine, or any combination of the three.

To learn which machine is the System Control machine, issue the **bos status** command on any server machine, using **upclient** as the argument for the **-process** option. The output for the **upclient** process used to obtain administrative lists from the System Control machine includes the **upclient** command used to start the process. The first parameter of the command is the name of the System Control machine; the second parameter is the pathname to the administrative lists on that machine; for example, *dcelocal*/**var/dfs**.

To learn which machine is a Binary Distribution machine, issue the **bos status** command on a server machine of the CPU/OS type you wish to check, again using **upclient** as the argument for **-process**. The output for the **upclient** process used to obtain binary files from the Binary Distribution machine includes the **upclient** command used to start the process. The first parameter of the command is the name of the Binary Distribution machine; the second parameter is the pathname to the binary files on that machine; for example, *dcelocal*/**bin**.

When using the **bos status** command to determine machine roles, always use the **-long** option to display more detailed information about the specified processes. You must use the **-long** option to determine the exact role of a server machine.

The following examples illustrate how to determine whether a machine is a System Control machine or a Binary Distribution machine. The output for a server machine that is neither a System Control machine nor a Binary Distribution machine displays that no **upserver** is running.

```
$ bos status /.../abc.com/fs1 upserver -long

bos: failed to get instance info for 'upserver' (no such entity)
```

The output for a System Control machine includes references to the **upserver** process and the *dcelocal*/**var/dfs** directory, where administrative lists are stored.

```
$ bos status /.../abc.com/fs2 upserver -long

Instance upserver, (type is simple) currently running normally.
Process last started at Mon Nov 4 05:23:54 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/upserver dcelocal/var/dfs'
```

The output for a Binary Distribution machine includes references to the **upserver** process and the *dcelocal*/**bin** directory, where binary files for processes and programs are stored.

```
$ bos status /.../abc.com/fs3 upserver -long

Instance upserver, (type is simple) currently running normally.
Process last started at Mon Nov 4 05:16:31 1991 (1 proc starts)
Parameter 1 is 'dcelocal/bin/upserver dcelocal/bin'
```

16.5 Stopping and Removing Processes

You can stop a process by using the **bos stop** command to set its status flag to **NotRun** in both the BOS Server's memory and the **BosConfig** file. The process then appears as `disabled` in the output from the **bos status** command. The entry remains in the file, but it does not run again until you issue the **bos start** command, which changes its status flag to **Run** in *both* the memory and the **BosConfig** file. You can also issue the **bos startup** command to run the process by changing its status flag *only* in memory.

To halt a process temporarily (for example, to perform maintenance or make alterations to a configuration), use the **bos shutdown** command to change the process's status in the BOS Server's memory to **NotRun**. The effect is immediate and remains until you again change the memory state or until the BOS Server restarts, at which time it consults the **BosConfig** file and sets the memory state to match the information in the file.

After you stop a process with the **bos stop** command, you can remove it from the **BosConfig** file with the **bos delete** command. It then no longer

appears in the output from the **bos status** command. You must use the **bos stop** command to stop a process (**simple** or **cron**) whose status is **Run** before you use the **bos delete** command to remove it from the **BosConfig** file. An error occurs if the status of a process being deleted is **Run** when the **bos delete** command is issued.

Caution: Do not temporarily stop a database server process on all machines simultaneously. This would make the database totally unavailable.

16.5.1 Stopping Processes by Changing Their Status Flags to NotRun

To stop processes by changing their status flags to **NotRun**, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be stopped. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos stop** command to stop the processes by changing their status flags in the **BosConfig** file and in the BOS Server's memory to **NotRun**:

   $ **bos stop -server** *machine* **-process** *server_process...* [**-wait**]

   The **-wait** option causes the command shell prompt to remain absent until the processes have stopped. If omitted, the prompt immediately returns, even if the processes have not yet stopped.

16.5.2 Stopping Processes Temporarily

To stop processes temporarily, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be stopped. If necessary, issue the **bos lsadmin** command to check.

2. Issue the **bos shutdown** command to stop the processes by changing their status flags in the BOS Server's memory to **NotRun**:

   $ **bos shutdown -server** *machine* [**-process** *server_process...*] [**-wait**]

   The **-process** *server_process* option specifies each server process that is to be stopped. Omit this option to stop all processes except the BOS Server.

   The **-wait** option causes the command shell prompt to remain absent until the processes have stopped. If omitted, the prompt immediately returns, even if the processes have not yet stopped.

16.5.3 Removing Processes from the BosConfig File

To remove processes from the **BosConfig** file, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine from which the process is to be removed. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos stop** command to stop the processes by changing their status flags in the **BosConfig** file and in the BOS Server's memory to **NotRun**; you must also do this for **cron** processes, even though they do not run continuously.

   $ **bos stop -server** *machine* **-process** *server_process...* [**-wait**]

   The **-wait** option causes the command shell prompt to remain absent until the processes have stopped. If omitted, the prompt immediately returns, even if the processes have not yet stopped.
3. Remove the processes from the **BosConfig** file with the **bos delete** command:

   $ **bos delete -server** *machine* **-process** *server_process...*

16.6 Starting Processes

When starting processes, you can use the **bos start** command to change their status flags to **Run** in both the **BosConfig** file and in the BOS Server's memory. You can also start processes that are temporarily disabled (processes that have a status of **Run** in the **BosConfig** file but a status of **NotRun** in memory) by using the **bos startup** command and changing only the memory state to **Run**. You can use the **bos startup** command to change a process's status in memory to **Run** even if its status in the **BosConfig** file is **NotRun**; thus, you can use the **bos startup** command to run tests on a server process without enabling it permanently.

A newly started process is a completely new instance; if you install new binaries during the time a process is shut down, they are used when you issue **bos start** or **bos startup**. If an instance of a process is already running, the only effect of these commands is to ensure that the process's status flag is set to **Run** in memory and, if **bos start** is used, in the **BosConfig** file, you must issue the **bos restart** command to start a new instance of the process.

16.6.1 Starting Processes by Changing Their Status Flags to Run

To start processes by changing their status flags to **Run**, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bos start** command to start the processes by changing their status flags in the **BosConfig** file and in memory to **Run**:

   ```
   $ bos start -server machine -process server_process...
   ```

16.6.2 Starting All Stopped Processes That Have BosConfig Flags of Run

To start all stopped processes that have status flags of **Run** in the **BosConfig** file, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Use the **bos startup** command to start each process that has a status flag of **Run** in the **BosConfig** file; this changes each process's status flag in the BOS Server's memory from **NotRun** to **Run**. Each process's status flag in the **BosConfig** file remains the same.

   $ **bos startup -server** *machine*

16.6.3 Starting Specific Temporarily Stopped Processes

To start processes that were temporarily stopped, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine where the processes are to be started. If necessary, issue the **bos lsadmin** command to check.
2. Use the **bos startup** command to start each specified process by changing its status flag in the BOS Server's memory to **Run**. Each process's status flag in the **BosConfig** file remains unchanged.

   $ **bos startup -server** *machine* **-process** *server_process...*

16.7 Restarting Processes

You may sometimes need to stop and then restart a process (for example, to load a new binary file immediately rather than wait for the BOS Server to perform its daily check for new files, which is described in Section 16.9.) You can use the **bos restart** command to stop and restart any or all processes on a server machine, including the BOS Server itself. The **bos restart** command can be used to restart only those processes already controlled by the BOS Server. It does not change the status flag for a process in the **BosConfig** file.

Caution: Restarting some processes can cause a service outage. You should schedule these outages for times of low usage on the system.

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the processes are to be restarted. If necessary, issue the **bos lsadmin** command to check.

2. There are three ways to use the **bos restart** command: you can stop and restart specific processes; you can stop and restart all processes, *including* the BOS Server; or you can stop and restart all processes *except* the BOS Server.

   To stop and restart specific processes, use the **-process** option with the **bos restart** command. Specify the name of each server process to be stopped and restarted. The BOS Server stops and immediately restarts all specified processes, regardless of their status flags in the **BosConfig** file.

   ```
   $ bos restart -server machine -process server_process...
   ```

   To restart all processes including the BOS Server, use the **-bosserver** option with the **bos restart** command. The BOS Server stops all processes, including itself. A new BOS Server immediately starts; it then restarts all processes with the status flag **Run** in the **BosConfig** file.

   ```
   $ bos restart -server machine -bosserver
   ```

The **-bosserver** option indicates that the BOS Server on **-server** is to stop all processes, including itself; a new BOS Server starts, restarting all processes with the status flag **Run**.

To stop and restart all processes except the BOS Server, omit both the **-process** and **-bosserver** options from the **bos restart** command. The BOS Server stops all processes except itself. It then immediately restarts all processes with the status flag **Run** in the **BosConfig** file.

```
$ bos restart -server machine
```

16.8 Installing Process Binary Files

Binary files for DFS server processes are stored on the local disk of each server machine. By default, the files are stored in the *dcelocal*/**bin** directory. The Binary Distribution machine for each CPU/OS type in a cell houses the master versions of the binary files for its machine and operating system type in this same local directory. The files can be stored in a different directory on any machine, but it avoids potential confusion if they are stored in the default directory on all machines.

The **bos install** command can be used to install a new process binary file on a server machine. It should be used to install new binary files only on Binary Distribution machines. The files are then distributed from each Binary Distribution machine to other server machines of the same CPU/OS type via the Update Server. By default, the **upclient** process on each server machine checks the Binary Distribution machine of its CPU/OS type for new (or different) versions of binary files every 5 minutes; if it finds that versions of files installed on the Binary Distribution machine are different from those on the local machine, it automatically copies the files to its local machine via the **upserver** process on the Binary Distribution machine.

Do not install new binary files on a server machine other than the Binary Distribution machine. The binary files are overwritten the next time the **upclient** process on the machine copies versions of files from the Binary Distribution machine of its CPU/OS type.

The **bos install** command preserves former versions of files in the installation directory by assigning **.BAK** and **.OLD** extensions as follows:

- If a current version of the file exists, the command adds a **.BAK** extension to its name.
- If a **.BAK** version of the file exists, the command changes its extension to **.OLD** before giving the current version a **.BAK** extension.
- If **.BAK** and **.OLD** versions of the file exist and the current **.BAK** version is less than 7 days old, the current version of the file overwrites the current **.BAK** version, but the **.OLD** version remains unchanged.
- If **.BAK** and **.OLD** versions of the file exist and the current **.BAK** version is at least 7 days old, the current **.BAK** version overwrites the current **.OLD** version, and the current version of the file overwrites the current **.BAK** version. Use the **bos getdates** command to examine the timestamps of current, **.BAK**, and **.OLD** versions of binary files to determine when they were installed.

The **bos install** command installs all files with the UNIX mode bits set to **755**, regardless of the mode bits associated with a version of the file that currently exists in the installation directory. These permissions are subject to the **umask** associated with the BOS Server on the machine on which the files are installed. The mode bits associated with the current version of the file are preserved when it becomes the **.BAK** version. The **bos install** command neither preserves nor manipulates the access control list, or ACL, permissions of a file installed from or to a DCE LFS fileset.

The **bos uninstall** command replaces the current version of a binary file with the next-oldest version of the file: the **.BAK** version, if it exists; otherwise, it is replaced with the **.OLD** version. If both the **.BAK** and **.OLD** versions exist, the **.OLD** version replaces the **.BAK** version when the latter becomes the current version. The **bos uninstall** command's **-all** option can be used to remove all versions of a binary file.

The **bos prune** command can be used to remove only the **.BAK** and **.OLD** versions of binary files from the *dcelocal*/**bin** directory. The **bos prune** command can also be used to remove core files, which are generated when processes monitored by the BOS Server go down, from the *dcelocal*/**var/dfs/adm** directory.

After new versions of binary files for processes controlled by the BOS Server are installed on a machine, you can use the **bos restart** command to begin using them immediately. Otherwise, the new versions are not used until the next new binary restart time (specified in the *dcelocal*/**var/dfs/BosConfig** file) of the BOS Server on the machine. (See Section 16.9 for detailed information about checking and setting scheduled restart times.)

16.8.1 Installing New Binary Files

To install new binary files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary files are to be installed. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos install** command to install a new version of each specified binary file:

   $ **bos install -server** *machine* **-file** *binary_file*... [**-dir** *alternate_dest*]

   The **-file** *binary_file* option specifies the pathname of each binary file to be installed on the machine specified with **-server**. For each file, specify either a full or a relative pathname; relative pathnames are interpreted relative to the current working directory. An installed file replaces a file of the same name.

   The **-dir** *alternate_dest* option specifies the pathname of the directory on **-server** in which all specified files are to be installed. Omit the **-dir** option to install the files in the default directory, *dcelocal*/**bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

16.8.2 Replacing Binary Files with Older Versions

To replace binary files with older versions of the files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary files reside. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos uninstall** command to replace the current version of each specified binary file with its next-oldest (**.BAK** or **.OLD**) version:

   $ **bos uninstall -server** *machine* **-file** *binary_file...* [**-dir** *alternate_dest*] [**-all**]

   The **-file** *binary_file* option specifies the name of each binary file to be replaced with its next-oldest version. Each specified file is replaced with its **.BAK** version, if it exists; otherwise, it is replaced with its **.OLD** version. If both the **.BAK** and **.OLD** versions exist, the **.OLD** version also replaces the **.BAK** version. All specified files must reside in the same directory (*dcelocal*/**bin** or an alternate directory specified with the **-dir** option). Specify only filenames; the command ignores all but the final component of a pathname.

   The **-dir** *alternate_dest* option provides the pathname of the directory in which all specified files reside. Omit the **-dir** option if the files reside in the default directory, *dcelocal*/**bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

   The **-all** option indicates that all versions (current, **.BAK**, and **.OLD**) of each specified file are to be removed.

16.8.3 Checking the Timestamps on Binary Files

Enter the **bos getdates** command to determine when binary files were installed:

$ **bos getdates -server** *machine* **-file** *binary_file...* [**-dir** *alternate_dest*]

The **-file** *binary_file* option specifies the name of the current version of each binary file whose timestamps are to be displayed. The timestamps on the current, **.BAK**, and **.OLD** versions of each file are displayed. All specified files must reside in the same directory, *dcelocal*/**bin**, or an alternate directory specified with the **-dir** option. Specify only filenames; the command ignores all but the final component of a pathname.

The **-dir** *alternate_dest* option specifies the pathname of the directory in which all specified files reside. Omit this option if the files reside in the default directory, *dcelocal*/**bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

16.8.4 Removing Old Binary and Core Files

To remove old binary and core files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary and core files reside. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos prune** command to remove old versions of **.BAK** files, **.OLD** files, core files, or any combination of the files from the *dcelocal*/**bin** and *dcelocal*/**var/dfs/adm** directories:

   $ **bos prune -server** *machine* [**-bak**] [**-old**] [**-core**] [**-all**]

   The **-bak** option specifies that all **.BAK** files are to be removed from *dcelocal*/**bin**. Use this option and optionally **-old**, **-core**, or both, or use **-all**.

   The **-old** option specifies that all **.OLD** files are to be removed from *dcelocal*/**bin**. Use this option and optionally **-bak**, **-core**, or both, or use **-all**.

   The **-core** option specifies that all core files are to be removed from *dcelocal*/**var/dfs/adm**. Use this option and optionally **-bak**, **-old**, or both, or use **-all**.

   The **-all** option specifies that all **.BAK** and **.OLD** files are to be removed from *dcelocal*/**bin** and all core files are to be removed from *dcelocal*/**var/dfs/adm**. Use this option or use some combination of **-bak**, **-old**, and **-core**.

16.8.5 Removing All Versions of Binary Files

To remove all versions of binary files, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine on which the binary files reside. If necessary, issue the **bos lsadmin** command to check.
2. Enter the **bos uninstall** command and include its **-all** option to remove all versions of each specified binary file:

   $ **bos uninstall -server** *machine* **-file** *binary_file...* [**-dir** *alternate_dest*] [**-all**]

   The **-file** *binary_file* option specifies the name of each binary file to be removed. The current, **.BAK**, and **.OLD** versions of each file are removed. All specified files must reside in the same directory, *dcelocal*/**bin**, or an alternate directory specified with the **-dir** option. Specify only filenames; the command ignores all but the final component of a pathname.

   The **-dir** *alternate_dest* option specifies the pathname of the directory in which all specified files reside. Omit this option if the files reside in the default directory, *dcelocal*/**bin**; otherwise, provide a full or relative pathname. Relative pathnames are interpreted relative to the *dcelocal* directory on **-server**.

   The **-all** option indicates that all versions (current, **.BAK**, and **.OLD**) of each specified file are to be removed.

16.9 Setting Scheduled Restart Times

The BOS Server performs two types of scheduled restarts: a general restart and a new binary restart. During a general restart, the BOS Server restarts itself (using a new binary file, if one exists) and then it restarts all other server processes on its machine that have a status flag of **Run** in the **BosConfig** file. It is recommended that the general restart time be set as a weekly time; by default, the BOS Server performs this type of restart weekly, on Sunday at 4:00 a.m.

During a new binary restart, the BOS Server checks for newly installed binary files. Binary files are installed on Binary Distribution machines with the **bos install** command, after which they are propagated to other machines by the Update Server. If a new version of a process's binary file was installed in *dcelocal*/**bin** after the process last started on the server machine, the BOS Server restarts the process so that the new instance of the binary file is used. It is recommended that the new binary restart time be specified as a daily time; by default, the BOS Server executes this type of restart daily at 5:00 a.m.

The default general and new binary restart times are set for early morning, when system usage is typically lowest. The **BosConfig** file on every server machine records the two restart times. This is a local file, so the information can be different for different machines. You can check or reset both time settings with the **bos getrestart** and **bos setrestart** commands.

A restart time can be set as a day and time or as just a time. When including a day, specify the day first, in lowercase letters; you can enter the entire name or just the first three letters; for example, **sunday** or **sun**. When indicating a time, separate hours from minutes with a colon; you can use 24-hour time or 1:00 through 12:00 with **am** or **pm**; for example, **14:30** or **"2:30 pm"**. You must enclose the entire entry in double quotes if it contains spaces; for example, **"2:30 pm"** or **"sun 14:30"**.

Caution: Never edit the restart times in the **BosConfig** file directly; use the **bos setrestart** command only. If you edit the restart times directly, the BOS Server does not recognize the new times until it is restarted and again reads the **BosConfig** file.

16.9.1 Checking the Current Restart Times

To check the current time settings, issue the **bos getrestart** command:

```
$ bos getrestart -server machine
```

Following is an example of the output from this command:

```
$ bos getrestart /.../abc.com/hosts/fs3

Server /.../abc.com/hosts/fs3 restarts at sun 4:00 am
Server /.../abc.com/hosts/fs3 restarts for new binaries at 5:00 am
```

16.9.2 Setting the General Restart Time

To set the general restart time, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine whose general restart time is to be set. If necessary, issue the **bos lsadmin** command to check.
2. Set the general restart time by issuing the **bos setrestart** command with the **-general** option:

   ```
   $ bos setrestart -server machine -general time
   ```

   The **-general** option identifies this as the general restart time, not the new binary restart time; *time* is the time at which the BOS Server is to restart itself and the other processes it controls.

16.9.3 Setting the New Binary Restart Time

To set the new binary restart time, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine whose new binary restart time is to be set. If necessary, issue the **bos lsadmin** command to check.
2. Set the new binary restart time by issuing the **bos setrestart** command with the **-newbinary** option:

   $ **bos setrestart -server** *machine* **-newbinary** *time*

   The **-newbinary** option identifies this as the new binary restart time, not the general restart time; *time* is the time at which the BOS Server is to check for new binary files in *dcelocal*/**bin**.

16.10 Rebooting a Server Machine

Note: Consult other DCE component documentation to determine the impact of rebooting on other DCE components.

Rebooting a server machine, while not difficult, should never be the first method used to solve DFS-related problems. You should reboot a server machine only if no other recourse is available, such as when a process that is not controlled by the BOS Server fails. Rebooting causes a service outage. If the machine being rebooted is the only Fileset Database machine, it can make the entire file system unavailable to all users; if the machine is a File Server machine, people using filesets located only on that machine (for example, user filesets) cannot access those filesets.

To prepare a server machine for powering down, you can issue the **bos shutdown** command to have the BOS Server shut down the other server processes that are running on the machine; the BOS Server does not shut itself down—it terminates correctly when you turn off the machine. You can then reboot the machine by issuing the machine's **reboot** command (or its equivalent).

You can reboot a machine from either the local console or the console of a remote machine (via **telnet** or an appropriate program). The two approaches are essentially the same, with the exception that rebooting from the local console lets you track the status of the reboot as it occurs, which you cannot do with remote rebooting. Regardless of the reboot method you use, server processes restart automatically after the reboot if

the machine's initialization file (**/etc/rc** or its equivalent) contains the following instruction to restart the BOS Server automatically:

dcelocal`/bin/bosserver`

1. *To reboot from the console of a remote machine*, open a remote connection to the machine you want to reboot (using **telnet** or an appropriate program). To reboot from the local console of the machine you want to reboot, omit this step.
2. Verify that you have the necessary privilege. You must be included in the **admin.bos** file on the machine to be rebooted. If necessary, issue the **bos lsadmin** command to check.
3. Issue the **bos shutdown** command to prepare to power down the machine to be rebooted. This command directs the BOS Server to shut down the other DFS server processes that are running on the machine by changing their status flags in the BOS Server's memory to **NotRun**. The BOS Server does not shut itself down; it terminates safely when you turn off the machine. Include the **-wait** option to be sure that all processes have stopped before performing the next step.

   `$` **bos shutdown -server** *machine* **-wait**

   The **-wait** option causes the command shell prompt to remain absent until the processes are stopped. If the **-wait** option is omitted, the prompt returns immediately, even if the processes are not yet stopped.
4. Log in as **root** in the native UNIX file system of the machine to be rebooted. For example:

   `$` **su root**
   `Password:` *root_password*

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

   # /etc/reboot

Chapter 17

Making Filesets and Aggregates Available

In the DCE Local File System (DCE LFS), a fileset is defined as a collection of related files that are organized into a single, easily managed unit. Because DCE LFS filesets are usually smaller in size than standard file systems, and because each DCE LFS aggregate can house multiple DCE LFS filesets, DCE LFS filesets are easily moved between File Server machines to facilitate load balancing across the network. It is also easy to place read-only copies (replicas) of DCE LFS filesets on different machines in your cell. These multiple copies prevent machines from becoming overburdened with requests for files from popular filesets.

In other operating systems, a file system typically occupies more disk space and is tied to a physical location. In addition, non-DCE LFS file systems (non-DCE LFS filesets) cannot be replicated in DFS.

This chapter provides detailed information about how to create, replicate, and back up DCE LFS filesets, and how to mount DCE LFS and non-DCE LFS filesets for use in the DCE namespace. It also explains how to export aggregates and partitions. (See Chapter 18 for information on the tasks involved in the use and maintenance of filesets.)

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

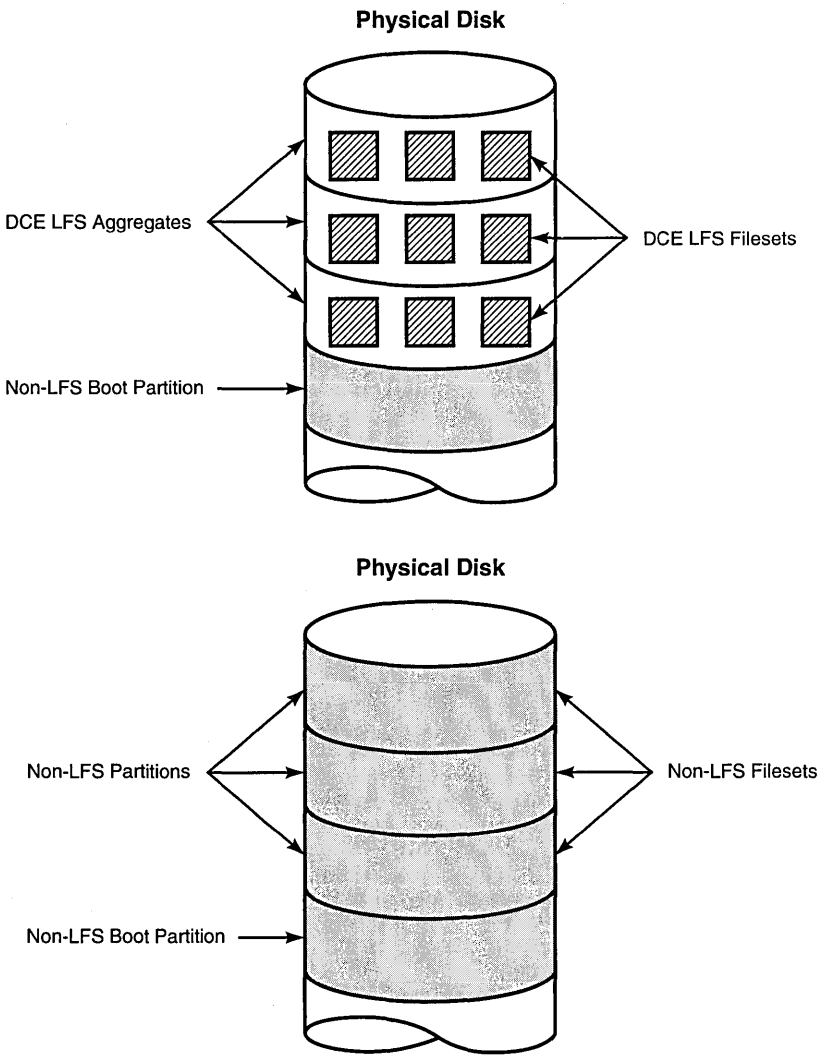
17.1 An Overview of Filesets

A DCE LFS fileset is a hierarchical grouping of files that is managed as a single unit. A DCE LFS aggregate is a disk partition that is modified to include the DCE LFS metadata structure that supports DCE Access Control Lists (ACLs), multiple DCE LFS filesets, logging, and other fileset-related operations.

Using DFS, you can share information stored on the local disks of different machines by exporting aggregates and partitions from the machines. Exporting an aggregate or partition makes the filesets contained on it available in the DCE namespace. With the DCE LFS, you can export multiple filesets from one aggregate. Because non-LFS partitions do not support the enhancements that are supported on DCE LFS aggregates, and because you can store only one fileset on a non-LFS partition, you can export only one non-LFS fileset per non-LFS partition.

Figure 17-1 illustrates the structural differences between the DCE LFS and other file systems. The partitioning structure in the DCE LFS features aggregates, each of which can store multiple DCE LFS filesets; the partitioning structure in other file systems has partitions that can house only a single non-LFS file system each. (Note that the disks in both structures include a non-LFS boot partition.)

Figure 17–1. Comparison of DCE LFS and non-LFS Disk Partitioning Structures



17.1.1 Creating and Using Filesets

Before you can create a DCE LFS fileset, you must first use the **newaggr** command to initialize, or format, the disk partition on which the aggregate is to reside; the **newaggr** command is comparable to the UNIX **newfs** command. You must then export the aggregate with the **dfsexport** command to make it available in the DCE namespace.

After the DCE LFS aggregate is initialized and exported, a DCE LFS fileset can be created on it with the **fts create** command. This command also registers the fileset in the Fileset Location Database (FLDB) and obtains a unique ID number for the fileset. To make the contents of a DCE LFS fileset visible in the DCE namespace, enter the **fts crmount** command to create a mount point for the fileset. After the **fts crmount** command is issued, the fileset is automatically attached to the DFS file system and is accessible to authorized DCE users.

On the other hand, when creating a non-LFS fileset for use on the local machine, you do not use **fts** commands. Because a disk partition is equal to one non-LFS fileset, the **newfs** command (or the command appropriate to your system) is used to initialize the partition on which the non-LFS fileset is to reside. The **mount** command (or its appropriate equivalent) is then used to mount the partition locally, after which data can be placed on the partition and used locally. Note that if your vendor has properly modified your local operating system's **mount** command, you can also mount and use a DCE LFS fileset locally.

To make a non-LFS fileset visible in the DCE namespace, you first use the **fts crfldbentry** command to register the fileset in the FLDB and generate a unique ID number for it. You then export the partition on which the fileset resides with the **dfsexport** command and mount the fileset with the **fts crmount** command. The terms *aggregate* and *non-LFS aggregate* can also be used to refer to an exported partition.

17.1.2 The Different Types of DCE LFS Filesets

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

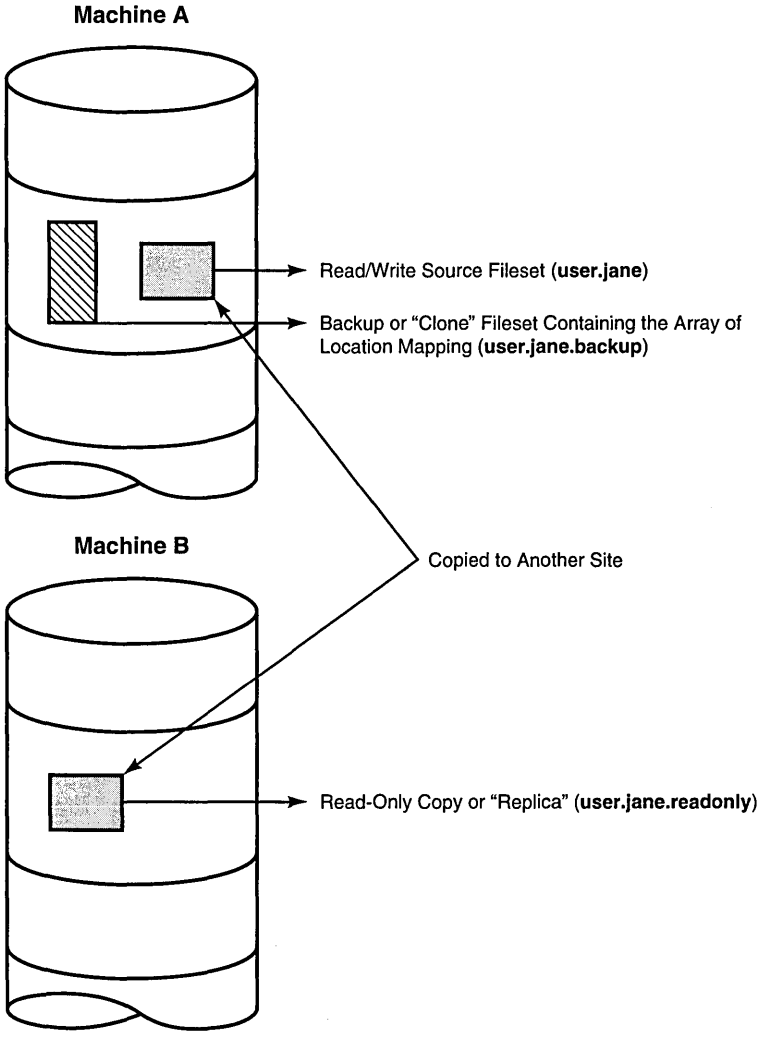
Every DCE LFS fileset has a single read/write version, which contains the modifiable versions of the files and directories in that fileset. This version is also referred to as the read/write source because the other fileset types are derived from it via replication and backup operations.

A read-only fileset is an exact copy, or replica, of all of the data in a read/write source fileset when the read-only replica is created. Each read-only fileset is given the same name as its read/write source with an additional **.readonly** extension. Read-only filesets can be placed at various sites in the file system; a site is a specific aggregate on a File Server machine. A read-only fileset cannot be modified by commands such as **mkdir** or **rm** (or their equivalent commands). If the read/write source fileset changes, the read-only versions must be updated to match the changed read/write version; otherwise, they remain unchanged. The update process can be performed manually (via Release Replication) or it can be automated (via Scheduled Replication).

A backup fileset is a clone of a read/write source fileset stored at the same site and with the same name as the source, with the addition of a **.backup** extension. A backup fileset copies the array that tracks the location of the data in a read/write source fileset; it does not copy the data in the source fileset. The size of a backup fileset grows in proportion to changes that are made to its read/write source. A backup fileset is not the same as a backup *of* a fileset (for example, a copy on tape), but making a backup fileset is often one step in the backup process. (See Chapter 20 for more information on the backup process.)

Figure 17-2 illustrates the different types of DCE LFS filesets: read/write, read-only, and backup.

Figure 17–2. The Different Types of DCE LFS Filesets



17.1.3 Identifying DCE LFS and Non-LFS Filesets

Every DCE LFS and non-LFS fileset is identified by a unique name and ID number. The following subsections discuss these two forms of fileset identification.

17.1.3.1 Fileset Names

Every fileset must have a fileset name that is unique within the cell in which it resides. The name is stored in the cell's FLDB. You assign a name to a DCE LFS fileset when you create the read/write version of the fileset and register it in the FLDB with the **fts create** command. You assign a name to a non-LFS fileset when you register the read/write (and only) version of the fileset in the FLDB with the **fts crfldbentry** command.

You can use the following characters in the name of a fileset:

- All uppercase and lowercase alphabetic characters (a through z, and A through Z)
- All numerals (0 through 9)
- The . (period)
- The - (dash)
- The _ (underscore)

A fileset name must include at least one alphabetic character or an _ (underscore); it cannot consist of just numbers, periods, and dashes. This allows the system to differentiate the name of the fileset from its ID number.

The name you assign to a fileset can contain no more than 102 characters. This does not include the **.readonly** or **.backup** extension, which is automatically added when a process creates a read-only or backup fileset.

Note: Fileset names can actually be as long as 111 characters—the name of the fileset plus the appropriate **.readonly** or **.backup** extension; however, you can specify only the first 102 characters of the name to accommodate the extensions. This is also true of non-LFS fileset names, even though non-LFS filesets do not need the extensions because they cannot have read-only and backup versions.

With DCE LFS, each user's home directory typically corresponds to a separate fileset. You may find it convenient to name all user filesets **user.***user_name* (for example, **user.sandy**). It may also be convenient to indicate the type of aggregate in which the fileset is stored (for example, **ufs.fs1** to indicate a non-LFS aggregate from the machine named **fs1**). You may also want to put system binaries into filesets with names that begin with the system type (for example, **osf1_pmax.bin**).

When specifying the name of an existing fileset in a DFS command, include the **.readonly** or **.backup** extension, if appropriate, to indicate the read-only or backup version of the fileset. You must include the extension when you wish to perform an operation that affects only that version of a fileset. For example, to delete just the backup version of a fileset, you must add the **.backup** extension to the name of the fileset when you issue the **fts delete** command.

17.1.3.2 Fileset ID Numbers

Every fileset also has a fileset ID number that, like a fileset name, is unique within the cell in which the fileset resides. When a fileset is registered in the FLDB with the **fts create** or **fts crfldbentry** command, the FL Server allocates it a fileset ID number, which is stored in the FLDB along with its name. Read/write and backup filesets have their own fileset IDs, which are automatically reserved in the FLDB when the read/write source fileset is registered; all read-only copies of the same read/write fileset share a common fileset ID.

Fileset ID numbers are represented as two positive integers separated by a pair of commas. For example, the ID number of the first fileset in the FLDB is **0,,1**. The integer after the commas is then incremented every time a new fileset is created. When the integer after the commas becomes larger than 2^{32}, the integer before the commas becomes 1 and the integer after the commas returns to 0 (zero).

When specifying a fileset ID in a DFS command, you can omit the integer before the commas if it is a 0 (zero); commands that accept a fileset ID number assume that the first integer is 0 (zero) if it is not supplied. In this case, also omit the two commas. For example, the fileset ID number **0,,1** can be entered as **1**.

17.1.4 Tracking Fileset Locations

The Fileset Location Database (FLDB) is maintained by the FL Server. The FLDB records information about the locations of the filesets in a cell. Users do not have to track the location of a fileset; the Cache Manager contacts the FL Server to obtain the location of a requested fileset.

Each read/write fileset has an entry in the FLDB; the entry includes information about the fileset's read-only and backup versions. Read-only and backup filesets normally do not have their own FLDB entries because they share an FLDB entry with the read/write fileset. However, a read-only fileset can have its own entry if its read/write source is removed.

For a DCE LFS fileset, information is also stored in the fileset's header, which is part of the data structure that records the physical addresses on the aggregate of the files in the fileset. It is essential that the FLDB entry and the fileset header be synchronized (that they match). All **fts** commands that affect fileset status and the FLDB change both the appropriate FLDB entry and the fileset header. In some rare cases, however, you may need to resynchronize the entries yourself, as described in Chapter 18.

A non-LFS fileset does not have a fileset header, but some information about the fileset is available from the local disk of the machine on which it resides. For example, the fileset ID number of each non-LFS fileset is stored in the *dcelocal*/**var/dfs/dfstab** on the local machine. (See Section 17.2 for a description of the **dfstab** file.)

17.1.4.1 A Fileset's FLDB Information

The **fts lsfldb** and **fts lsft** commands display information from a fileset's FLDB entry (the **fts lsft** command also shows information from a fileset's header). Each FLDB entry for a DCE LFS fileset contains the following information:

- The name of the fileset, with a **.readonly** or **.backup** extension, if appropriate.
- The fileset IDs of the read/write, read-only, and backup versions.
- A separate status flag for each of the three versions, indicating whether the version exists at some site. A status of `valid` indicates the version

exists at some site; a status of `invalid` indicates the version does not exist at any site. Note that the status of the read-only version is `valid` once a replication site is defined, regardless of whether a replica yet exists at the site.

- The number of sites at which a version of the fileset exists.
- An indicator if the FLDB entry is locked. The indicator is omitted if the entry is not locked.
- The replication parameters that are associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) where read/write (`RW`), read-only (`RO`), or backup (`BK`) versions of the fileset reside.
- For each read-only site, the MaxSiteAge replication parameter defined for that site. (See Section 17.4 for more information about replication parameters.)
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine in the FLDB or `<nil>` if no group owns the server entry.

Because functionality, such as replication, is not supported for non-LFS filesets, FLDB entries for non-LFS filesets do not contain as much information as entries for DCE LFS filesets do. However, information, such as the ID number and site of the fileset, is recorded in the FLDB. The **fts lsfldb** and **fts lsft** commands display this information.

17.1.4.2 A Fileset's Header Information

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

The **fts lsheader** and **fts lsft** commands display information from a fileset's header (the **fts lsft** command also shows information from a fileset's entry in the FLDB). Each fileset header for a DCE LFS fileset contains the following information:

- The name of the fileset, with a **.readonly** or **.backup** extension, if appropriate.
- The fileset ID number.
- The type of fileset (`RW` for read/write, `RO` for read-only, or `BK` for backup).
- The storage space used by the fileset, in kilobytes.
- Additional internal information about the fileset and its access status.
- The status flag for the site, including `On-line`, `Off-line`, or an error condition.
- The File Server machine, aggregate name, and aggregate ID number where the fileset resides. This information, while not in the header, is available because it was used to contact the machine that houses the fileset.
- The ID numbers of the parent, clone, and backup filesets that are related to the fileset.
- The ID numbers of the low-level backing and low-level forward filesets that are related to the fileset.
- The version number of the fileset. Every DCE LFS fileset has a distinct version number that increments every time an operation is performed on the fileset or a file it contains. Version numbers have the same format as fileset ID numbers (for example, **0,,25963**).
- A quota, indicating the maximum amount of disk space the fileset is permitted to occupy.
- The day, date, and time that the fileset was created (for read-only and backup versions, this indicates the day, date, and time that the fileset was replicated or backed up).
- The day, date, and time that the contents of the fileset were last updated.

Because non-LFS filesets do not have DCE LFS fileset headers, only such information as the fileset ID number is available from the machine that houses the fileset. The **fts lsheader** and **fts lsft** commands display this information.

17.1.5 Replicating DCE LFS Filesets

Replication is the process of creating one or more read-only copies of the read/write version of a DCE LFS fileset and placing the copies at multiple sites. With replication, you can tailor your system's configuration by making a fileset's contents accessible from more than one File Server machine. As a result, a single machine is not overburdened with requests for popular files, and files are available from more than one machine in the event of machine failure. Replication is not available for non-LFS filesets.

You can manually initiate the replication of a DCE LFS fileset by using Release Replication, or you can automate the process by using Scheduled Replication. With Release Replication, you issue a command that updates the read-only copies whenever you want to release new read-only copies of the read/write fileset. This type of replication is useful for filesets whose replication you want to control closely. With Scheduled Replication, you specify parameters that control how often read-only copies are updated. The Replication Server then updates the copies at the specified intervals. This type of replication is useful for filesets whose replication is to be performed asynchronously.

To be able to use read-only versions of a fileset, you must also create read-only versions of all filesets that are mounted above it at higher levels in the file system. In other words, you must create read-only copies of the filesets that contain the fileset's parent directories. When the Cache Manager traverses a pathname to locate a file, it begins at the top-level fileset in your cell and accesses the read-only versions of filesets whenever possible. However, if any fileset in the pathname has no read-only version, the Cache Manager accesses the read/write version and never accesses a read-only version for as long as it traverses the remainder of the pathname.

17.1.6 Mounting Filesets

In order to make a DCE LFS or non-LFS fileset's contents visible and accessible to users in the DCE namespace, the fileset is attached to the namespace through a mount point. In DFS, you use the **fts crmount** command to create a mount point for a fileset. There are several types of mount points; the tasks in this chapter all use regular mount points, which are the most common type. (See Section 17.6 for more information about the other types of mount points.)

A DFS mount point appears and functions like a regular directory, but structurally it is a special symbolic link that indicates the name of the fileset associated with a mount point. The fileset has a directory structure whose root directory has the same name as the mount point. You can create standard subdirectories within the fileset's root directory. You can also create other mount points there, which appear like subdirectories; these mount points are then associated with files in their own filesets rather than with files in the mount-level directory's fileset.

Each exported fileset in DFS is mounted automatically once a mount point is created for it; you do not have to issue additional commands to attach a fileset. (See Section 17.6 for more about mount points and how they are accessed.)

17.1.7 Standard Options and Arguments

When using the **fts** commands to create, manipulate, or delete filesets, you can often add the **-verbose** option to receive detailed information from the **fts** program about its actions as it executes the command. This can be particularly helpful if an operation fails for a reason you do not understand. The amount of additional information produced by the **-verbose** option varies for different commands.

The following options and arguments are common to many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete details about each command.)

- The **-fileset** *name* option is the complete name (for example, **user.sandy**) or ID number (for example, **0,,34692**) of the fileset to be used in the command.
- The **-server** *machine* option is the File Server machine to be used in the command. Use the DCE pathname (for example, **/.../abc.com/hosts/fs1**) with all commands unless otherwise indicated.
- The **-aggregate** *name* option is the device name (for example, **/dev/lv01**), aggregate name (for example, **lfs1** or **/usr**), or aggregate ID (for example, **3** or **12**) of the aggregate or partition to be used in the command. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal*/**var/dfs/dfstab** file.
- The **-cell** *cellname* option specifies the cell with respect to which the command is to be run (for example, **abc.com**). The default is the local cell of the issuer of the command.
- The **-noauth** option directs the **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See Chapter 15 for information about disabling DFS authorization checking.) If you use this option, do not use the **-localauth** option.
- The **-localauth** option directs the **fts** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server** (for example, **/.../abc.com/hosts/fs1/dfs-server**). Do not confuse a machine's DFS server principal with its unique **self** identity. (See Section 17.2 for information about DFS server principals.)

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

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

17.2 Exporting Aggregates and Partitions

Before exporting a DCE LFS aggregate or a non-LFS partition (non-LFS aggregate) from a File Server machine, you must ensure that an RPC binding exists for the DCE pathname of the machine and that a corresponding DFS server principal exists for the machine. You must also ensure that a server entry exists for the machine. The RPC binding is created in CDS, the DFS server principal is registered in the Registry Database, and the server entry is registered in the FLDB. (See Section 17.2.1 for a description of these prerequisites and additional requirements for exporting.)

Prior to exporting a DCE LFS aggregate, you must use the **newaggr** command to construct the aggregate from a raw disk partition. This command formats the partition for use as a DCE LFS aggregate. It is similar to the UNIX **newfs** command, which is used to format a partition. You issue it once for each DCE LFS aggregate that you want to create.

The partition to be initialized as a DCE LFS aggregate must be neither mounted locally nor exported to the DCE namespace when you issue the **newaggr** command. Conversely, before exporting a non-LFS partition for use as an aggregate in DFS, you must create the partition and mount it locally using the **newfs** and **mount** commands or their equivalents, and you must create an entry for the partition in the local **fstab** file or its equivalent.

To make data on a DCE LFS aggregate or non-LFS partition available in the DCE namespace, you must issue the **dfsexport** command to export the aggregate or partition. Before using the **dfsexport** command, include an entry in the *dcelocal*/**var/dfs/dfstab** file for each aggregate or partition to be exported. The **dfsexport** command reads the **dfstab** file to determine which aggregates and partitions can be exported. It then exports the indicated devices.

When the **dfsexport** command reads the **dfstab** file, it copies the entry for each exported aggregate or partition to the *dcelocal*/**var/dfs/dfsatab** file; it creates the **dfsatab** file if the file does not already exist. This file maintains a list of all currently exported aggregates and partitions. The **dfsexport** command will not export an aggregate or partition that already has an entry in the **dfsatab** file.

You typically add the **dfsexport** command to a machine's initialization file (**/etc/rc** or its equivalent) to automatically export aggregates and partitions

at system startup. A command that removes the **dfsatab** file usually appears in the initialization file before the **dfsexport** command.

Because a non-LFS partition can store only one fileset, you register that fileset in the FLDB with the **fts crfldbentry** command *before* you export the partition to the namespace. Using the **fts crfldbentry** command, you specify a name to be associated with the fileset; the FL Server allocates a fileset ID number for the new fileset. You use this fileset ID number when you create the entry for the partition in the **dfstab** file. After the partition is exported, you use the **fts crmount** command to create a mount point for the fileset the partition contains. The **fts crmount** command makes a fileset visible in the DCE namespace.

Conversely, the **dfsexport** command must be used to export a DCE LFS aggregate to the DCE namespace *before* the aggregate can store filesets. Once a DCE LFS aggregate is exported, filesets can be created on it with the **fts create** command, and mount points can be created for the filesets with the **fts crmount** command. You specify a name for a DCE LFS fileset with the **fts create** command; the fileset is automatically assigned an ID number and registered in the FLDB.

The following subsections describe these steps and the commands that are used to perform them in more detail. (See the DCE DFS portion of the *OSF DCE Administration Reference* for more information on a specific command.)

17.2.1 Preparing for Exporting

Several prerequisites must be met before you can export either a DCE LFS aggregate or a non-LFS partition from a File Server machine. The following server processes must be running before any of the other steps described in this or the following subsections are attempted:

- A CDS server process (**cdsd**) must be running in the cell, and a CDS advertiser process (**cdsadv**) and a CDS clerk process (**cdsclerk**) must be running on the machine; use the appropriate CDS command to verify that the CDS processes are running. Note that the CDS advertiser on a machine automatically spawns a new CDS clerk for each user on the machine.

- A Security Server process (**secd**) must be running in the cell, and a security client process (**sec_clientd**) must be running on the machine; use the appropriate Security Service command to verify that the processes are running.
- An RPC process (**rpcd**) must be running on the machine; use the appropriate RPC command to verify that the RPC process is running.
- An FL Server process must be running in the cell. You can use the **bos status** command to verify that the **flserver** process is running, or you can use the **fts lsfldb** command to list information about the **root.dfs** fileset, which must exist, thus verifying that the **flserver** is actually functioning.

In addition, an RPC binding must exist for the DCE pathname of the File Server machine in CDS, a corresponding DFS server principal must exist for the machine in the Registry Database, and a server entry must exist for the machine in the FLDB. The following subsections describe these topics and additional preparation that must be completed if the machine is to export aggregates or partitions.

17.2.1.1 Creating an RPC Binding for a File Server Machine

A File Server machine must have an RPC binding in CDS for its DCE pathname. The **fts** program uses the RPC binding to contact the File Server machine when it needs to communicate with the **ftserver** process on the machine. This RPC binding includes the server machine's network address, an identifier for the protocol used to communicate with the machine, and an endpoint for communications with the **rpcd** process on the machine.

In the DCE, server machines are identified by DCE pathnames of the form /.../*cellname*/**hosts**/*hostname*; for example, /.../**abc.com/hosts/fs1**. The entry in CDS for the RPC binding defined for each server machine must have a name of the form /.../*cellname*/**hosts**/*hostname*/**self**; for example, /.../**abc.com/hosts/fs1/self**.

Note that the DFS server principal for the machine is similarly derived. The abbreviated server principal registered for the machine in the FLDB is similar in form to the RPC binding and DFS server principal.

Note: The element that follows the *cellname* in the pathname is not well known; for example, **hosts** could be **dfs-hosts**. However, the string used for the element must be applied consistently to all such names in the cell.

To create an RPC binding for a File Server machine, use the **rpccp** and **cdscp** commands to create the structure for the RPC binding in CDS. The entry in CDS for the RPC binding must have a name of the form */.../cellname/***hosts**/*hostname/***self**.

17.2.1.2 Creating a DFS Server Principal for a File Server Machine

A File Server machine must also have a DFS server principal registered in the local Registry Database. The DFS server principal name is used to establish an authenticated connection to the DFS server machine.

In the DCE, server machines are identified by DCE pathnames of the form */.../cellname/***hosts**/*hostname*; for example, **/.../abc.com/hosts/fs1**. The DFS server principal is of the form */.../cellname/***hosts**/*hostname/***dfs-server**; for example, **/.../abc.com/hosts/fs1/dfs-server**. A machine's DFS server principal is similar in appearance to the name of its RPC binding, the difference being that the last element of the RPC binding name is **self**, whereas the corresponding element of the DFS server principal is **dfs-server**. The two elements also differ in that the RPC binding is defined in CDS, while the DFS server principal is registered in the Registry Database. Note again that **hosts** is not a well-known element of the name.

An abbreviation of the DFS server principal registered in the Registry Database must be used as the principal name associated with the machine's entry in the FLDB. Continuing with the previous example, **hosts/fs1** is the abbreviated DFS server principal associated with the FLDB entry for the machine whose DFS server principal in the Registry Database is **/.../abc.com/hosts/fs1/dfs-server**. (The full DFS principal name of a server machine is also associated with a server encryption key in a keytab file; see Chapter 15 for more information on server encryption keys.)

To create a DFS Server principal for a File Server machine, use the **rgy_edit** command to create a DFS server principal in the Registry Database for the machine from which aggregates and partitions are to be

exported. The DFS server principal must be of the form */.../cellname*/**hosts**/*hostname*/**dfs-server**.

17.2.1.3 Creating a Server Entry for a File Server Machine

Before it can house an exported aggregate or partition, a File Server machine must also have a server entry in the FLDB. The **fts crserverentry** command is used to register a File Server machine's server entry in the FLDB. The server entry stores information about the machine, such as its network addresses (a server entry can store up to four network addresses), its abbreviated DFS server principal name, the number of fileset entries in the FLDB that can be associated with it, and the group of administrators that ''owns'' (possesses special administrative privileges for) the server entry.

The **-principal** option of the **fts crserverentry** command is used to specify the abbreviated DFS server principal to be registered with a machine's server entry. The abbreviated DFS server principal is of the form **hosts**/*hostname*. For example, the DFS server principal **/.../abc.com/hosts/fs1/dfs-server** would be abbreviated to **hosts/fs1** for use with the machine's server entry in the FLDB.

The **-quota** option of the **fts crserverentry** command is used to limit the number of filesets (read/write, read-only, and backup) that can reside on a File Server machine. The entry for each fileset in the FLDB defines the File Server machine on which each version of the fileset resides. Each server entry records the total number of filesets that are listed in fileset entries as residing on the File Server machine. No more than the number of filesets that are specified with the **-quota** option can be recorded in the FLDB as residing on the machine at any given time.

The **-owner** option of the command is used to specify the group that owns the server entry. Members of this group can administer the FLDB entries for all filesets on the File Server machine. The administrators in the group need not be included on the **admin.fl** list for the entire cell, which would allow them to modify all of the fileset entries in the FLDB in that cell. The same group can be given ownership of the server entries for all of the File Server machines in an administrative domain, which is a collection of File Server machines administered by the same system administrators. Members of the group can then manipulate the FLDB entries for all of the filesets in the domain.

The following additional commands are also provided for the manipulation of server entries in the FLDB: the **fts lsserverentry** command lets you list current server entries; the **fts edserverentry** command allows you to edit an existing server entry; and the **fts delserverentry** command lets you remove an existing server entry. The following subsections provide information about the various server entry manipulation commands.

17.2.1.3.1 Creating a Server Entry for a Machine

To create a server entry for a File Server machine, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.fl** file on each Fileset Database machine. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Use the **fts crserverentry** command to create a server entry in the FLDB for the machine:

   $ **fts crserverentry -server** {*machine* | *address*} **-principal** *name* [**-quota** *entries*] [**-owner** *group*]

   The **-server** *machine* or *address* option specifies the DCE pathname or network address of the server machine. The command fails if a network address in use by another server entry is specified with this option.

   The **-principal** *name* option is the abbreviated DFS server principal name of the machine to be registered in the FLDB. The machine's principal name in the Registry Database must match this name.

   The **-quota** *entries* option sets a limit on the number of fileset entries (read/write, read-only, and backup) in the FLDB that can be associated with the server. If this option is omitted, the default is 0 (zero), meaning that an unlimited number of fileset entries can be associated with the server.

   The **-owner** *group* option specifies the name of the group that is the owner of the server entry. A group from the local cell can be specified by a full or an abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name. If this option

is omitted, no group owns the server entry; the value <nil> is reported as the owner.

17.2.1.3.2 Listing Server Entries for Machines

Use the **fts lsserverentry** command to list either the server entry for a specific File Server machine or all current server entries from the FLDB:

$ **fts lsserverentry** {**-server** *machine* | **-all**}

The **-server** *machine* option specifies the DCE pathname of the server machine whose entry in the FLDB is to be displayed. Use this option or use the **-all** option.

The **-all** option specifies that the entries for all server machines in the FLDB are to be displayed. Use this option or use the **-server** option.

17.2.1.3.3 Editing a Server Entry for a Machine

To edit the server entry for a File Server machine, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.fl** file on each Fileset Database machine. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Use the **fts edserverentry** command to modify any aspect of an existing server entry in the FLDB. For example, the command can be used to add an additional network address to an existing entry for a machine.

   $ **fts edserverentry -server** {*machine* | *address*}
   [{**-rmaddr** | **-addaddr** *address* | **-changeaddr** *address*}]
   [**-principal** *name*] [**-quota** *entries*] [{**-owner** *group* | **-noowner**}]

   The **-server** *machine* or *address* option specifies the DCE pathname or network address of the server machine whose entry in the FLDB is to be modified. Specify the network address if the **-rmaddr**, **-addaddr**, or **-changeaddr** option is used with the command.

The **-rmaddr** option removes the network address specified with **-server** from the FLDB. The command fails if the specified address is the only address present for the machine in the FLDB. If you use this option, do not use the **-addaddr** or **-changeaddr** option.

The **-addaddr** *address* option adds the additional address specified with this option to the FLDB for the machine specified with **-server**. A machine can have up to four addresses associated with its entry in the FLDB. If you use this option, do not use the **-rmaddr** or **-changeaddr** option.

The **-changeaddr** *address* option changes the address in the FLDB specified with **-server** to the address specified with this option. If you use this option, do not use the **-rmaddr** or **-addaddr** option.

The **-principal** *name* option changes the abbreviated DFS server principal name of the machine registered in the FLDB. The machine's principal name in the Registry Database must match this name. Omit this option to leave the DFS server principal registered for the machine unchanged.

The **-quota** *entries* option changes the limit on the number of fileset entries (read/write, read-only, and backup) in the FLDB that can be associated with the server. Omit this option to leave the quota for the number of fileset entries unchanged.

The **-owner** *group* option changes the group that is the owner of the server entry. In the entry, the specified group replaces the current owning group, if there is any. A group from the local cell can be specified by a full or an abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name. Use this option or use the **-noowner** option; omit both options to leave the current owning group unchanged.

The **-noowner** option specifies that no group is to own the server entry. In the entry, the empty group ID, which is displayed as `<nil>`, replaces the group that currently owns the server entry; the entry is unchanged in this regard if no group presently owns the server entry. Use this option or use the **-owner** option; omit both options to leave the current owning group unchanged.

17.2.1.3.4 Deleting a Server Entry for a Machine

To remove the server entry for a File Server machine, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.fl** file on each Fileset Database machine. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Use the **fts delserverentry** command to delete the server entry for a machine from the FLDB. The command fails if the entry in the FLDB for any fileset references the server entry to be removed as the location of the fileset.

   $ **fts delserverentry -server** {*machine* | *address*}

   The **-server** option specifies the DCE pathname or network address of the server machine whose entry is to be removed. The command fails if a fileset entry references the server entry to be removed.

17.2.1.4 Preparing a File Server Machine for Exporting

The following additional prerequisites must be met before a File Server machine can begin to export aggregates or partitions:

- The BOS Server (**bosserver** process) must be running on the machine.
- A keytab file and a server encryption key must exist on the machine.
- The **dfsbind** process must be running on the machine.
- The **fxd** process must be running on the machine.
- The Fileset Server (**ftserver** process) must be running on the machine.
- The Replication Server (**repserver** process) must be running on the machine, if the machine is to house read-only DCE LFS filesets.

The following procedure provides instructions for starting these processes and generating a key. The instructions assume that the **rgy_edit** command has already been used to create a keytab file on the machine.

1. Log in as **root** on the machine.

2. Start the BOS Server (**bosserver** process) on the machine with the **bosserver** command, using the **-noauth** option to disable DFS authorization checking on the server machine. (See Chapter 15 for a thorough description of DFS authorization checking.) The process automatically creates the **admin.bos** file when it starts.

   # bosserver -noauth

   The **-noauth** option starts the **bosserver** with DFS authorization checking turned off.

3. Use the **bos addadmin** command to add the necessary administrative users and groups to the **admin.bos** file. Make sure you are included in the list of users or groups added to the list. You must use the **-noauth** option to use the identity **nobody** as the identity of the issuer of the command.

   # bos addadmin -server *machine* **-adminlist admin.bos** [**-principal** *name...*] [**-group** *name...*] **-noauth**

   The **-adminlist admin.bos** option specifies that principals and groups are to be added to the **admin.bos** list on the machine indicated with the **-server** option.

   The **-principal** *name* option specifies the principal name of each user to be added to the **admin.bos** list. A user from the local cell can be specified by a full or an abbreviated principal name (for example, */.../cellname/username* or just *username*); a user from a foreign cell can be specified only by a full principal name.

   The **-group** *name* option specifies the name of each group to be added to the **admin.bos** list. A group from the local cell can be specified by a full or an abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

4. Add a server encryption key to the keytab file on the machine with the **bos genkey** command, again using the **-noauth** option. (See Chapter 15 for complete details about managing a keytab file.)

   # bos genkey -server *machine* **-kvno** *version_number* **-noauth**

   The **-kvno** *version_number* option is the key version number of the new key. Valid arguments for this option are decimal integers from 0 (zero) to 255.

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

5. Enable DFS authorization checking on the machine with the **bos setauth** command, once again using the **-noauth** option.

   # bos setauth -server *machine* **-authchecking on -noauth**

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

   The **-noauth** option directs the **bos** program to use the unprivileged identity **nobody** as the identity of the issuer.

6. Start the **dfsbind** process on the machine.

   # dfsbind

7. Start the **fxd** process to initialize the File Exporter in the kernel of the machine. Specify the name of the proper administrative group with the **-admingroup** option. (See Chapter 14 for more information about creating administrative groups.)

   # fxd -admingroup *group*

   The **-admingroup** *group* option specifies the group that can administer the File Exporter on the machine. A group from the local cell can be specified by a full or an abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

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

9. Start the Fileset Server (**ftserver** process) with the **bos create** command. (See Chapter 16 for complete information about starting a server process.) The **admin.ft** file is created automatically when the process starts.

   $ **bos create -server** *machine* **-process ftserver -type simple**
   -cmd */dcelocal*/**bin/ftserver**

   The **-process ftserver** option specifies that the process to be created and started is to be identified by the name **ftserver**.

   The **-type simple** option specifies that the **ftserver** process is to be a **simple** process.

   The **-cmd** */dcelocal*/**bin/ftserver** option provides the full pathname to the binary file for the **ftserver** process.

10. Use the **bos addadmin** command to add the necessary administrative users and groups (and possibly server machines) to the **admin.ft** file.

    $ **bos addadmin -server** *machine* **-adminlist admin.ft** [**-principal** *name...*]
    [**-group** *name...*]

    The **-adminlist admin.ft** option specifies that principals and groups are to be added to the **admin.ft** list on the machine indicated with the **-server** option.

    The **-principal** *name* option specifies the principal name of each user or server machine to be added to the **admin.ft** list. A principal from the local cell can be specified by a full or an abbreviated principal name (for example, */.../cellname/username* or just *username*); a principal from a foreign cell can be specified only by a full principal name.

    The **-group** *name* option specifies the name of each group to be added to the **admin.ft** list. A group from the local cell can be specified by a full or an abbreviated group name (for example, */.../cellname/group_name* or just *group_name*); a group from a foreign cell can be specified only by a full group name.

11. Start the Replication Server (**repserver** process) with the **bos create** command. No administrative list is associated with the **repserver** process.

    $ **bos create -server** *machine* **-process repserver -type simple -cmd** *dcelocal*/**bin/repserver**

    The **-process repserver** option specifies that the process to be created and started is to be identified by the name **repserver**.

    The **-type simple** option specifies that the **repserver** process is to be a **simple** process.

    The **-cmd** */dcelocal*/**bin/repserver** option provides the full pathname to the binary file for the **repserver** process.

12. After the Fileset Server process is started, use the **fts statftserver** command to verify that the process is performing requested actions. This command is useful mainly if you believe the process is not functioning properly.

    $ **fts statftserver -server** *machine*

    The **fts statftserver** command displays the message `No active transactions on` *machine* if the Fileset Server is functioning properly. It displays additional information if the Fileset Server is currently performing an action. Depending on the information displayed, the Fileset Server may or may not be functioning properly.

17.2.2 Exporting DCE LFS Aggregates

The following subsections introduce and describe the steps that are involved in initializing and exporting a DCE LFS aggregate. Before exporting a DCE LFS aggregate to the DCE namespace, the prerequisites described in the previous section must be met: the necessary Directory Service, Security Service, RPC, and DFS server processes must be running; an RPC binding must exist for the DCE pathname of the machine; a DFS server principal must exist for the machine; a server entry must exist for the machine; and a keytab file and a key must exist on the machine. You must also initialize the aggregate by formatting it with the **newaggr** command before it can be exported.

17.2.2.1 An Overview of Initializing DCE LFS Aggregates

Prior to creating a DCE LFS aggregate, use the **newaggr** command to initialize the raw partition on which the aggregate is to reside by formatting it for use as a DCE LFS aggregate. The **newaggr** command creates the metadata structure used by the DCE LFS for ACL support, logging, multiple fileset storage, and other fileset-related operations. It also allocates temporary space for use by the DCE LFS log for faster restarts after system failures. The DCE LFS log is not a file; it is a structure that resides on an aggregate. If the system fails, the logged metadata that was written to disk is replayed at system restart to return the system to a consistent state.

Because the **newaggr** command overwrites all data on the partition being initialized, the partition being initialized should not contain data you want to save when the command is issued. Also, the command fails if the partition or aggregate being initialized is currently exported to the DCE namespace. It also fails if the aggregate to be initialized houses a locally mounted fileset. Finally, if the partition is mounted locally, the **newaggr** command causes the kernel to panic. (Note that a non-LFS partition must be mounted locally before it can be exported.)

If you are uncertain about which arguments to supply with the **newaggr** command, execute the command with the **-noaction** option. This option directs the command to report on what it would do without actually modifying the partition. When using the **-noaction** option, supply the other options as you would when actually executing the command.

In operating systems that support logical volumes, the **newaggr** command can be used to initialize a logical volume as a DCE LFS aggregate. In such cases, all of the command's functionality described here with respect to a disk partition applies to the logical volume.

17.2.2.2 Initializing a DCE LFS Aggregate

To initialize a DCE LFS aggregate, do the following:

Caution: Do not use the **newaggr** command to initialize a non-LFS partition, especially if it contains data you want to retain, unless you want to convert the partition to a DCE LFS aggregate; the command destroys all data on the specified partition. Also, do not use the command on a locally mounted partition; doing so causes the kernel to panic. Finally, do not use the command on a partition or aggregate that is currently exported to the DCE namespace or on an aggregate that houses a locally mounted fileset; the command fails in these cases.

1. Log in as **root** on the machine on which the new aggregate is to be initialized.
2. Issue the **newaggr** command to initialize the aggregate. (See the DCE DFS portion of the *OSF DCE Administration Reference* for more detailed information about the **newaggr** command.)

   # **newaggr -aggregate** *name* **-blocksize** *bytes* **-fragsize** *bytes* [**-initialempty** *blocks*] [**-aggrsize** *blocks*] [**-logsize** *blocks*] [**-overwrite**] [**-verbose**] [**-noaction**]

   The **-aggregate** *name* option is the device name or aggregate name of the disk partition to be initialized as a DCE LFS aggregate. These identifiers are specified in the first and second fields of the entry for the aggregate in the *dcelocal*/**var/dfs/dfstab** file.

   The **-blocksize** *bytes* option is the number of bytes to be available in each DCE LFS block on the aggregate. Allowable values are the powers of 2 from 1024 to 65,536.

   The **-fragsize** *bytes* option is the number of bytes to be available in each DCE LFS fragment on the aggregate. Allowable values are the powers of 2 from 1024 to the number of bytes specified with the **-blocksize** option.

   The **-initialempty** *blocks* option is the number of DCE LFS blocks to be left empty at the beginning of the partition when the aggregate is initialized. Allowable values are from 0 (zero) to 65,536 divided by

the number of bytes specified with the **-blocksize** option; for example, if 65,536 is specified with the **-blocksize** option, the **-initialempty** option can be 0 or 1. If this option is omitted, one block is left empty.

The **-aggrsize** *blocks* option is the total number of DCE LFS blocks that are to be available on the aggregate. Because this value cannot exceed the size of the partition, it can be used only to restrict the size of the aggregate. It must be large enough to accommodate at least the log and any blocks left empty at the beginning of the partition. If this option is omitted, the size of the partition being initialized is used.

The **-logsize** *blocks* option is the number of DCE LFS blocks to be reserved for the log on the aggregate. This value cannot exceed the number of DCE LFS blocks used for the **-aggrsize** option, and it must specify at least enough blocks for the log to be initially created. If this option is omitted, 1% of the total number of DCE LFS blocks on the aggregate (**-aggrsize**) is used.

The **-overwrite** option specifies that an existing file system found on the partition can be overwritten. If this option is omitted and a file system is found on the partition, the command informs you that a file system already exists. It then terminates with an exit code of at least 16 without overwriting the existing file system.

The **-noaction** option directs the command to display information about what it would do without actually modifying the partition. Include the other options as you would to actually execute the command. The command displays the default values it would use for its options and informs you if the partition already contains a file system.

17.2.2.3 Exporting a DCE LFS Aggregate

To export a DCE LFS aggregate, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine from which the aggregate is to be exported, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine from which the aggregate is to be exported. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directories in which the mount points for any filesets are to be created. If necessary, issue the **acl_edit -l** command to check the ACL permissions for the directories. Note that you need to have the **w** and **x** permissions for any directories in non-LFS filesets.
3. Log in as **root** on the machine from which the aggregate is to be exported.
4. Use a text editor to edit the **dfstab** file to include an entry for the DCE LFS aggregate to be exported. The following fields appear for each entry in the file, in the order listed. Each field must be separated by a minimum of one space or tab; each entry must be on a separate line.
   - Device Name: The block device name of the aggregate (for example, **/dev/lv03**).
   - Aggregate Name: The name to be associated with the exported aggregate. An aggregate name can contain any characters, but it can be no longer than 31 characters, and it must be different from any other aggregate name in the file. Aggregate names cannot be abbreviated, so you should choose a short, explicit name (for example, **lfs1**).
   - File System Type: The identifier for the file system type of the aggregate. For DCE LFS aggregates, this must be **lfs**. It must be in lowercase letters.
   - Aggregate ID: A positive integer to act as the aggregate ID of the exported aggregate. The integer must be different from any other aggregate ID in the **dfstab** file. (If the ID is changed after the aggregate contains one or more filesets, fileset operations on those filesets will fail.)

   The following entry from a **dfstab** file is for a DCE LFS aggregate:

```
/dev/lv03  lfs1  lfs  3
```

5. Issue the **dfsexport** command to export the aggregate to the DCE namespace. Before exporting, this command reads the **dfstab** file to determine which aggregates and partitions are available to be exported. Omit all of the command's options to list the aggregates and partitions 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 the **-all** option with the **-type** option to export only DCE LFS aggregates or only non-LFS partitions. Use this option or use the **-aggregate** option.

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

   The **-type** *name* option is the file system type to be exported. Specify **lfs** to export only DCE LFS aggregates; specify **ufs** to export only non-LFS partitions. Use the **-type** option only with the **-all** option (it is ignored if it is used without **-all**); omit **-type** and use **-all** to export all aggregates and partitions.

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

7. Issue the **fts create** command to create a DCE LFS fileset on the aggregate and register the fileset in the FLDB. The FL Server allocates a unique fileset ID number for the fileset. (See Section 17.3 for detailed information about DCE LFS fileset creation.)

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

   The **-ftname** *name* option is the complete name to be associated with the fileset being created. The name can contain no more than 102 characters, and it must contain at least one alphabetic character or an _ (underscore).

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

8. Enter the **fts crmount** command to create a mount point in the file system for the new DCE LFS fileset. This makes the contents of the fileset visible to other users. (See Section 17.6 for more information about mounting filesets.)

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

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

   Repeat the **fts crmount** command for each fileset created in the previous step.

17.2.3 Exporting Non-LFS Partitions

This section describes the steps that are involved in exporting a non-LFS partition; after it is exported, a non-LFS partition can be referred to as a non-LFS aggregate. Before exporting a non-LFS partition to the DCE namespace, the prerequisites described in Section 17.2.1 must be met: the necessary CDS, Security Service, RPC, and DFS server processes must be running; an RPC binding must exist for the DCE pathname of the machine; a DFS server principal must exist for the machine; a server entry must exist for the machine; and a keytab file and a key must exist on the machine. You must also have created and locally mounted the partition (using the **newfs** and **mount** commands or their equivalents), and you must have listed the partition's name in the local **fstab** file (or its equivalent).

To export a non-LFS partition, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine from which the partition is to be exported. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the mount point for the fileset is to be created. If necessary, issue the **acl_edit -l** command to check

the ACL permissions for the directory. Note that you need to have the **w** and **x** permissions if the directory is in a non-LFS fileset.

3. Provide a name for the fileset (non-LFS file system) on the partition to be exported and register the fileset in the FLDB with the **fts crfldbentry** command. The number specified with the **-aggrid** option is also used as the partition's aggregate ID in the **dfstab** file; it must not already be in use in the **dfstab** file.

   The FL Server allocates a unique fileset ID number for the partition's lone non-LFS fileset. The **fts crfldbentry** command returns this ID number, along with two additional ID numbers allocated for read-only and backup versions of the fileset, even though a non-LFS fileset cannot have these versions. Use the read/write ID number returned by the command as the fileset ID in the **dfstab** file.

   ```
   $ fts crfldbentry -ftname name -server machine -aggrid ID
   ```

   The **-ftname** *name* option is the complete name to be associated with the fileset being registered. The name can contain no more than 102 characters, and it must contain at least one alphabetic character or an _ (underscore).

   The **-aggrid** *ID* option is a positive integer to serve as the aggregate ID for the partition to be exported. The number must not already be in use in the **dfstab** file on the machine where the partition resides.

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

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

   - Device Name: The block device name of the partition; for example, **/dev/lv02.**
   - Aggregate Name: The name to be associated with the exported partition. The aggregate name of a non-LFS partition must match the name of its local mount point (for example, **/usr**). An aggregate name can contain any characters, but it can be no longer

than 31 characters, and it must be different from any other aggregate name in the file. Aggregate names cannot be abbreviated, so you should choose a short, explicit name.

- File System Type: The identifier for the file system type of the partition. For non-LFS file systems, this must be **ufs**. It must be in lowercase letters.
- Aggregate ID: A positive integer to serve as the aggregate ID of the exported partition. The integer must match the aggregate ID specified with the **-aggrid** option of the **fts crfldbentry** command, and it must be different from any other aggregate ID in the **dfstab** file. (If the ID is changed, fileset operations on the partition's fileset will fail.)
- Fileset ID: The unique fileset ID number returned by the **fts crfldbentry** command for the fileset on the partition (for example, **0,,18756**). Use the read/write ID number, not the read-only or backup ID number, returned by the command as the value for this field.

The following entry from a **dfstab** file is for a non-LFS partition:

```
/dev/lv02  /usr  ufs  1  0,,18756
```

6. Issue the **dfsexport** command to export the partition to the DCE namespace. Before exporting, this command reads the **dfstab** file to determine which aggregates and partitions are available to be exported. Omit all of the command's options to list the aggregates and partitions 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 the **-all** option with the **-type** option to export only DCE LFS aggregates or only non-LFS partitions. Use this option or use the **-aggregate** option.

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

The **-type** *name* option is the file system type to be exported. Specify **lfs** to export only DCE LFS aggregates; specify **ufs** to export only non-LFS partitions. Use the **-type** option only with the **-all** option (it is ignored if it is used without **-all**); omit **-type** and use **-all** to export all aggregates and partitions.

7. Log out as **root** from the machine to return to your authenticated DCE identity.
8. Enter the **fts crmount** command to create a mount point in the file system for the new non-LFS fileset. This makes the contents of the fileset visible to other users.

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

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

17.2.4 Exporting Aggregates and Partitions at System Startup

To export DCE LFS or non-LFS aggregates at system startup, use a text editor to edit the initialization file (**/etc/rc** or its equivalent) for the File Server machine to include the following DFS commands:

- The **dfsbind** command.
- The **fxd** command to start the File Exporter.
- The **dfsexport** command with the **-all** option to export all partitions and aggregates with entries in the **dfstab** file. Typically, the file includes the **rm** command to remove the **dfsatab** file before it executes the **dfsexport** command.
- The **bosserver** command to start the BOS Server. If the *dcelocal*/**var/dfs/BosConfig** file includes the recommended entries, this also starts the Fileset Server and the Replication Server on the machine.

17.2.5 Removing Aggregates and Partitions from the Namespace

If you exported a DCE LFS aggregate or non-LFS partition (either by adding the **dfsexport** command to the proper initialization file—**/etc/rc** or its equivalent—or by issuing the command directly), you can issue the **dfsexport** command with the **-detach** option to remove the aggregate or partition from the DCE namespace. When it successfully detaches an aggregate or partition, the **dfsexport** command removes the entry for the aggregate or partition from the **dfsatab** file.

Before the command detaches an exported aggregate or partition, it first revokes all of the tokens for data on the aggregate or partition. When its tokens are revoked, a client flushes the data that is cached from the aggregate or partition, writing any modified data back to the File Server machine. The command does not perform the detach operation if it cannot revoke all necessary tokens; it instead reports that the device is busy. In this case, the command's **-force** option can be included to force completion of the detach operation even if all necessary tokens cannot be revoked.

In general, avoid detaching an aggregate or partition if users are still accessing filesets that reside on it. The **dfsexport** command revokes all tokens for data on the aggregate or partition before it detaches it, but users accessing data from the aggregate or partition will not be able to save the data. Be especially cautious when using the **-force** option, which forces an aggregate or partition to be detached even if all tokens cannot be revoked.

17.2.6 Using DCE LFS Filesets Locally

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

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

Mounting a DCE LFS fileset locally does not improve local access time to the data on the fileset if the fileset is accessed via the DCE 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 that is physically located on the local disk of your machine is available only through the DCE namespace, a network outage makes it impossible to access the data. 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 a fileset is exported and the fileset is mounted in the DCE namespace, you can also access a locally mounted DCE LFS fileset globally. The pathname associated with the fileset is different for local and global access. For example, a fileset that is accessed in the local operating system as **/lfs/jlw** may be accessed in the DCE namespace as **/.../abc.com/fs/usr/jlw**.

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

Note: A DCE LFS fileset that is mounted locally cannot be moved to a different File Server machine (with the **fts move** command), and it cannot be deleted (with the **fts delete** or **fts zap** command); you must unmount it locally before attempting any of these operations. Also, an aggregate that houses a locally mounted fileset cannot be recovered or salvaged with the **salvage** command, nor can it be reinitialized with the **newaggr** command.

1. Log in as **root** on the machine. 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 to locally mount 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 startup by including the proper information in an initialization file (**fstab** or its equivalent). Refer to your vendor's documentation for more information about mounting file systems at system startup.

17.3 Creating Read/Write DCE LFS Filesets

Read/write DCE LFS filesets are created with the **fts create** command. The **fts create** command is used to create only DCE LFS filesets; non-LFS filesets are created by exporting and mounting non-LFS partitions (as described in Section 17.2.3).

Before creating a read/write DCE LFS fileset, select a site for the fileset. A site is an aggregate on the File Server machine where the fileset is to reside. If necessary, issue the **fts aggrinfo** command to make sure the aggregate you choose has enough space to accommodate the fileset. (See Chapter 18 for a detailed description of the **fts aggrinfo** command.)

You specify a fileset's name when you create it with the **fts create** command. A fileset's name should describe the fileset's contents; for example, the name **user.terry** describes a fileset that contains data for the user **terry**. A fileset name must also be unique within the local cell. It can be no greater than 102 characters in length, and it must contain at least one

alphabetic character or an _ (underscore). (See Section 17.1.3.1 for more information on fileset naming rules.)

The **fts create** command

- Creates a single FLDB entry for the read/write fileset and for any potential read-only and backup versions of the fileset.
- Allocates a fileset ID number for the read/write fileset. It also reserves ID numbers for the read-only and backup versions of the fileset in anticipation of their creation.
- Assigns the name that you specify to the fileset.
- Sets the FLDB site flag for the read/write fileset to `valid`, and sets the site flags for the read-only and backup filesets to `invalid` because they do not yet exist.
- Creates a fileset header at the site File Server machine and aggregate that you designate as the location of the fileset.
- Creates an empty root directory in the fileset. This directory becomes visible when the **fts crmount** command is used to mount the fileset.
- Records null ACLs as the default for use by the root directory of the fileset. Note that due to the interaction between UNIX mode bits and ACLs, the directory has a set of implicit initial ACLs that grant permissions to different users and groups. (See Chapter 14 for information about the interaction between ACLs and UNIX mode bits and for suggestions for initial ACLs.)
- Assigns a default quota of 5000 kilobytes to the fileset.

The following subsections describe the steps involved in creating a read/write DCE LFS fileset. Once a read/write fileset is created, you can create read-only and backup copies of the read/write fileset; the following sections provide detailed information about creating read-only and backup DCE LFS filesets. Note that the **fts create** command is also used to create a fileset for use as the top-level fileset for diskless machines.

17.3.1 Creating a Read/Write Fileset

After creating a read/write DCE LFS fileset with the **fts create** command, use the **fts crmount** command to create a mount point for the fileset. The mount point makes the contents of the fileset visible in the DCE namespace. You can use the **fts setquota** command to alter the fileset's default quota of 5000 kilobytes, and you can use the **acl_edit** command to modify its default ACLs.

17.3.1.1 Creating and Mounting a Read/Write Fileset

To create and mount a read/write fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to reside, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset is to reside. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the fileset is to be mounted. If necessary, issue the **acl_edit -l** command to check the permissions for the directory.
3. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate on which the fileset is to be created. (See Chapter 18 for a detailed description of the **fts aggrinfo** command.)

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

4. Enter the **fts create** command to create the fileset:

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

   The **-ftname** *name* option is the complete name to be associated with the fileset being created.

5. Enter the **fts crmount** command to create a mount point in the file system for the new fileset. This makes the contents of the fileset visible to other users.

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

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

17.3.1.2 Resetting the Fileset Quota

To reset the quota for the new read/write fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of the list.
2. Issue the **fts setquota** command to change the fileset's quota:

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

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

   The **-size** *kbytes* option is the maximum amount of disk space the fileset can occupy. Specify the value in kilobytes; a value of 1024 kilobytes equals 1 megabyte.

17.3.2 Creating Read/Write Filesets for Diskless Machines

The **fts create** command is also used to create a fileset for use as the top-level diskless fileset. After the fileset is created, the **fts crmount** command is used to mount the top-level diskless fileset in the DCE namespace. Because this fileset is to serve as the root directory for diskless machines, you need to copy common binary and configuration files (such as those found in the **/bin** and **/etc** directories or their equivalents) to the fileset.

You also need to mount the root of the DCE global namespace under the top-level diskless directory to enable the diskless machine to access the DCE namespace. The **-global** option is used with the **fts crmount** command to mount the root of the DCE namespace under the top-level diskless directory. Note that the **-global** option was not displayed with previous examples of the **fts crmount** command.

The DCE Diskless Support Service is used to enable the fileset to function as the top-level diskless directory. (See Part 3 of this guide for more information on the DCE Diskless Support Service.)

To create a read/write fileset to serve as the top-level fileset for diskless machines, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to reside, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset is to reside. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the fileset is to be mounted. If necessary, issue the **acl_edit -l** command to check the permissions for the directory.
3. Issue the **fts create** command to create the fileset that is to serve as the top-level fileset for diskless machines:

   ```
   $ fts create -ftname name -server machine -aggregate name
   ```

   The **-ftname** *name* option is the complete name to be associated with the fileset being created (for example, **diskless.root.osf1_pmax**).

4. Enter the **fts crmount** command to create a mount point in the DCE namespace for the fileset that is to serve as the top-level diskless fileset:

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

   The **-dir** *directory_name* option is the location for the root directory of the fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace (for example, **/.../abc.com/fs/diskless.root/osf1_pmax**). Include a complete pathname unless you want to mount the fileset in the working directory.

5. Copy any necessary files (for example, those files found in the **/bin** and **/etc** directories or their equivalents) to the diskless fileset with the appropriate command (**cp** or its equivalent).

6. Enter the **fts crmount** command to create a mount point for the root of the DCE global namespace under the top-level diskless fileset:

   $ **fts crmount -dir** *directory_name* **-global**

   The **-dir** *directory_name* option is the location for the root of the DCE namespace; the specified location must not already exist. However, the parent directory of the mount point must exist in the DCE namespace (for example, **/.../abc.com/fs/diskless.root/osf1_pmax/...**).

   The **-global** option indicates that the mount point is for the root of the DCE namespace.

7. Use the command appropriate for your system to make the fileset the top-level directory for the diskless machine.

17.4 Creating Read-Only DCE LFS Filesets

Replication is the process of creating read-only copies (replicas) of a read/write DCE LFS fileset and placing the copies on multiple File Server machines. Replication increases the availability of the fileset in the event of a network or server outage. If one of the machines that houses the fileset becomes unavailable, the fileset can still be accessed from another machine. Replication is not available for non-LFS filesets.

Replicate read/write filesets that match the following criteria:

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

The following two types of replication are available for DCE LFS filesets:

- Release Replication requires you to issue the **fts release** command to explicitly update the read-only versions of the read/write source fileset. The command places a read-only copy of the source fileset on the same File Server machine as the source. The Replication Server (**repserver** process) on each machine that houses a read-only replica then replaces the copy of the replica on its machine with the new replica stored on the same machine as the read/write fileset. The read-only replicas do not change until you explicitly issue the **fts release** command to update them.

  Use Release Replication if the fileset seldom changes or if you need to closely track the distribution of new replicas.

- Scheduled Replication requires you to specify replication parameters that control the length of time between the automatic release of new replicas. The Replication Server on each machine that houses a read-only site updates the read-only replica on its machine according to time intervals that you supply.

  Use Scheduled Replication if you prefer to have the system release updates of replicas at regular intervals and you do not need to closely track the distribution of new replicas.

A read/write fileset can be replicated via only one of the two types of replication at any one time. Regardless of the type of replication in use, an immediate update of the read-only copies of a source fileset can be requested at any time with the **fts update** command. The command can be used to update all replicas or only the replica at a specific site. It has no effect on the replication type and parameters that are defined for the read/write fileset.

The type of replication to be used for a fileset is set or changed with the **fts setrepinfo** command. This command is also used to set the replication parameters to be used with the fileset. After it is set, replication information is stored in the FLDB entry for the fileset.

Some replication parameters are used with both Release and Scheduled Replication to specify how the replicated data is to be used by Cache Managers; other parameters are used only with Scheduled Replication to define how often Replication Servers are to check for updated versions of the source fileset. An additional site-specific parameter used with Scheduled Replication is set with the **fts addsite** command. (See Section 17.4.4.1 for more information on these parameters.)

Each replica of a read/write fileset resides at a specified replication site (a specific File Server machine and aggregate). Before read-only versions of a fileset can be made, the **fts addsite** command must be used to define the sites where the read-only replicas are to reside. The site definitions are recorded in the entry for the fileset in the FLDB. It is not possible to place multiple read-only copies of the same fileset on a single File Server machine.

Although read-only replicas do not reside at the replication sites until the source fileset is first replicated, the status flag of the read-only version in the FLDB is changed from `invalid` to `valid` once a replication site is defined. It is assumed that replicas will be placed at the sites shortly after the sites are defined.

All read-only copies of a read/write fileset share the same fileset ID number. The number, which is reserved in the FLDB when the read/write fileset is

created, is one greater than the ID number of the read/write fileset. The fileset ID number of a read-only fileset's source fileset is referred to as the replica's parent ID number.

With Release Replication, the **fts addsite** command must be used to define a replication site on the File Server machine on which the source fileset resides before any subsequent replication sites can be added. The site for the replica can be defined on any aggregate on the source's File Server machine. However, it is best to define the site for this replica on the same aggregate as that on which the source fileset resides, in which case the replica is created as a clone of the source fileset. Because it is created as a clone fileset, which has the same structure as a backup fileset, a replica defined on the same aggregate as the source requires potentially much less space than a full read-only replica created on a different aggregate. (See Section 17.5 for more information about the structure of backup filesets.)

Note: Replication is available in a cell only if the following are true: **root.dfs**, the cell's main read/write fileset, is a DCE LFS fileset; **root.dfs** was mounted with an explicit read/write mount point when the cell was configured; and **root.dfs** is replicated. (See the *OSF DCE Administration Guide—Introduction* for information on configuring **root.dfs** to support replication.)

17.4.1 Preparing for Replication

The following prerequisites must be met before you can replicate a read/write fileset:

- Each File Server machine that is to house a replica of the fileset must have a server entry in the FLDB. (See Section 17.2.1.3 for information on creating server entries.)
- A Replication Server (**repserver** process) and a Fileset Server (**ftserver** process) must be running on each File Server machine that is to house a replica of the fileset. Use the **bos status** command to verify that the **repserver** and **ftserver** processes are running on the machine at each replication site. (You can also use the **fts statrepserver** command, which is described in Section 17.4.3, to verify the status of the Replication Server on a machine.)

- You must use the **fts setrepinfo** command to indicate the type of replication and to set the replication parameters to be used with the fileset that is to be replicated. This information is recorded in the FLDB entry for the fileset.
- You must use the **fts addsite** command to add the replication sites to the FLDB entry for the fileset. If necessary, enter the **fts aggrinfo** command to check the available space on a File Server machine's aggregates before defining replication sites on that machine. Use the **fts lsft** command to check the size of the read/write fileset. (See Chapter 18 for a description of the **fts aggrinfo** and **fts lsft** commands.) Remember, a read-only replica must also be stored on the same File Server machine as the read/write fileset with Release Replication.

The following subsections describe how to set the replication type and parameters and how to add replication sites.

17.4.1.1 Replication Type and Parameters

Before defining any replication sites, you must use the **fts setrepinfo** command to indicate the type of replication to be used and to set the replication parameters for the fileset to be replicated. When initially defining the type of replication to be used with a fileset, include either the **-release** option or the **-scheduled** option with the command to indicate the type of replication to be used.

Most of the options available with the **fts setrepinfo** command determine the replication parameters to be associated with the fileset; the primary exceptions are the **-change** and **-clear** options, which are described later in this section. Replication parameters define the time intervals used by Cache Managers on client machines that are accessing data from the read-only replicas and Replication Servers on File Server machines that house the replicas. Descriptions of the replication parameters follow; each parameter is set with an option of the same name. (Note that the term *replication parameter* is used to refer to the *replication intervals* that are defined for a fileset with the **fts setrepinfo** command, not to the parameters of the command.)

The following parameters apply to *both* Release Replication and Scheduled Replication:

- MaxAge specifies the maximum amount of time the Cache Manager distributes data it has obtained from a read-only replica without verifying that the data is still current with respect to the read/write version of the fileset.
- FailAge determines how long the Cache Manager continues to provide data it has obtained from a read-only replica before it fails to do so because the data is outdated. The Cache Manager fails to use cached data if the data is older than this value and the Cache Manager cannot obtain more current data.
- ReclaimWait specifies the amount of time the File Exporter waits before it reclaims storage space from deleted files (those not referenced by any directory). It also determines the frequency of the Cache Manager's keep-alive messages to the Replication Server.

  The Cache Manager sends keep-alive messages to indicate that it is still using files from a read-only replica. A file that is being accessed from a replica remains available as long as the Cache Manager continues to notify the Replication Server that the file is still in use and the Replication Server continues to forward these notifications to the File Exporter. This is true even if the file has been removed from all directories on the read/write fileset in the interim. The Cache Manager sends keep-alive messages more frequently than the ReclaimWait interval to prevent the File Exporter from reclaiming storage space that was occupied by deleted files.

The following parameters apply *only* to Scheduled Replication:

- MinRepDelay specifies how long the Replication Server waits after a read/write fileset changes before it attempts to get a new copy of the fileset. The Replication Server tracks the currency of replicas by maintaining a whole-fileset token for each fileset. If a Cache Manager changes the read/write fileset, the Replication Server relinquishes its whole-fileset token and waits for at least the time specified by MinRepDelay before requesting a new whole-fileset token.

- MaxSiteAge controls the maximum amount of time that a replica can be out of date. The Replication Server attempts to keep a replica current within this amount of time. A MaxSiteAge value is stored with each site; the MaxSiteAge for a site is set with the **fts addsite** command.
- DefaultSiteAge is the default value to be used as the MaxSiteAge for a replication site if that value is not set with the **fts addsite** command.

Table 17-1 summarizes the six parameters just described. For each parameter, the table describes the command with which the parameter is set, its default value, its usage (if any) with Release Replication, and its usage with Scheduled Replication; usage descriptions include details on the dependencies between the different parameters. Refer to this table when using the **fts setrepinfo** (or **fts addsite**) command to specify the replication parameters for a fileset (or site).

Note: Unless it is absolutely necessary to change them, it is recommended that you use the default parameters (with the exception of MaxAge) that are listed in the table.

Table 17–1. Descriptions of Replication Parameters

Parameter	Default	Release Replication	Scheduled Replication
MaxAge (**fts setrepinfo**)	2 hours	Required only if FailAge is specified	Required only if FailAge, MinRepDelay, or DefaultSiteAge is specified
FailAge (**fts setrepinfo**)	1 day or twice MaxAge, whichever is larger	Optional	Required only if MinRepDelay or DefaultSiteAge is specified
ReclaimWait (**fts setrepinfo**)	18 hours	Required only if FailAge is specified	Required only if FailAge, MinRepDelay, or DefaultSiteAge is specified
MinRepDelay (**fts setrepinfo**)	5 minutes or one-quarter of DefaultSiteAge, whichever is smaller	Not applicable	Required only if FailAge or DefaultSiteAge is specified
MaxSiteAge (**fts addsite**)	DefaultSiteAge	Not applicable	Required only if DefaultSiteAge is *not* specified
DefaultSiteAge (**fts setrepinfo**)	One-quarter of MaxAge	Not applicable	Optional

The system uses the guidelines listed in Table 17-1 to calculate default values for each of the parameters *unless* you specify

- FailAge for Release Replication
- FailAge, MinRepDelay, or DefaultSiteAge for Scheduled Replication

Once you specify one of these parameters, the system no longer performs any default calculations; you must specify values for all applicable parameters. The exception is DefaultSiteAge for Scheduled Replication, which is always optional. However, if the other parameters specified with the **fts setrepinfo** command are supplied, the system does not calculate a default value for DefaultSiteAge; you must specify the MaxSiteAge for each replication site with the **fts addsite** command. Also, because the

MinRepDelay, MaxSiteAge, and DefaultSiteAge parameters do not apply to Release Replication, they are recorded but otherwise ignored if they are specified for a fileset that uses Release Replication. (Note that they are used if the fileset's style of replication is ever changed to Scheduled Replication.)

17.4.1.1.1 Setting Replication Type and Parameters

To set a fileset's replication type and parameters, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

   Note: If you use the command's **-change** option to change a fileset's existing replication type from Release to Scheduled, you must also be included in the **admin.ft** file on the machine on which the read/write fileset resides if a replica actually resides at the replication site on that machine. (The first replication site defined for a fileset that uses Release Replication must be on the same File Server machine as the read/write fileset.)

2. Enter the **fts setrepinfo** command to set the replication parameters for the read/write fileset that is to be replicated. To use the default parameters described in Table 17-1, enter only the fileset name or ID number and the replication type. To specify one or more of the parameters, refer to the table for information on which parameters apply for the two types of replication and the dependencies between the parameters.

   Enter *interval* values as integers with the following abbreviations to designate units: **d** for days, **h** for hours, **m** for minutes, and **s** for seconds. For example, to indicate 3 days and 2 hours, enter **3d2h**. At least one of the four values (days, hours, minutes, or seconds) must be provided, and the unit abbreviations (**d**, **h**, **m**, or **s**) must be used with any integer. The unit abbreviations can be uppercase or

lowercase letters, and they can be entered in any order; for example, **3m2h** is a valid entry.

$ **fts setrepinfo -fileset** {*name* | *ID*} {**-release** | **-scheduled**} [**-change**] [**-maxage** *interval*] [**-failage** *interval*] [**-reclaimwait** *interval*] [**-minrepdelay** *interval*] [**-defaultsiteage** *interval*] [**-clear**]

The **-fileset** *name* or *ID* option is the complete name or ID number of the read/write DCE LFS fileset to be replicated.

The **-release** option specifies that Release Replication is to be used with the fileset. When initially setting the replication type, use this option or use the **-scheduled** option.

The **-scheduled** option specifies that Scheduled Replication is to be used with the fileset. When initially setting the replication type, use this option or use the **-release** option.

The **-change** option is used with **-release** or **-scheduled** to indicate that the replication type is to be changed. When changing the replication type for a fileset, you must include the **-change** option.

The **-maxage** *interval* option is the value for MaxAge. An effective value must be greater than or equal to 2 minutes.

The **-failage** *interval* option is the value for FailAge. An effective value must be greater than or equal to **-maxage**.

The **-reclaimwait** *interval* option is the value for ReclaimWait. An effective value must be greater than 2 hours; do not specify a value less than 90 minutes.

The **-minrepdelay** *interval* option is the value for MinRepDelay. This value must be less than the MaxSiteAge specified for each replication site with the **-maxsiteage** option of the **fts addsite** command.

The **-defaultsiteage** *interval* option is the value for DefaultSiteAge. This value is used with the **fts addsite** command if the **-maxsiteage** option is omitted from that command.

The **-clear** option removes all replication parameters previously defined for the fileset.

17.4.1.1.2 Changing Replication Type and Parameters

You can use the **fts setrepinfo** command to change the type of replication or the replication parameters that are associated with a fileset at any time after they are set. Brief descriptions of the operations used to change these values follow:

- *To change the replication parameters*, use the options for the parameters you want to change to indicate the new parameters.
- *To change all replication parameters*, use the **-clear** option to remove all previous replication parameters, and either use the options for the parameters you want to change to indicate the new parameters or omit the options to allow the system to calculate new replication parameters.
- *To change the replication type*, use the **-release** or **-scheduled** option to indicate the new type of replication to be used, and use the **-change** option to indicate that the type is being changed. Although not required, you may also want to use the **-clear** option to clear all previous replication parameters and then reset them using parameters that are better suited to the new type of replication.

  Because Scheduled Replication imposes more constraints than Release Replication, Release Replication does not require a replication site to have a MaxSiteAge. Therefore, it is likely that one or more Release Replication sites will have a MaxSiteAge of 0 (zero), which is the default value recorded for a site if no MaxSiteAge or DefaultSiteAge is specified. When changing from Release Replication to Scheduled Replication, the **-defaultsiteage** option *must* be used to set a DefaultSiteAge if any replication site does not have a MaxSiteAge and no DefaultSiteAge exists for the source fileset; otherwise, the **fts setrepinfo** command fails. If the command fails for this reason, reissue it, specifying a DefaultSiteAge with the **-defaultsiteage** parameter.

(See Section 17.4.1.1.1 for details about the syntax of the **fts setrepinfo** command. See Table 17-1 for information about which of the command's options apply to the two types of replication and the dependencies between the parameters.)

17.4.1.2 Adding and Removing Replication Sites

After you define the replication parameters that are associated with a read/write fileset, you must define the sites (File Server machines and aggregates) where read-only replicas of the fileset are to be stored. Use the **fts addsite** command to add a replication site for a read/write fileset. If you define an incorrect site or decide at some later time that you no longer want a replica to be stored at a specific site, you can use the **fts rmsite** command to remove a replication site for a fileset.

Note that you can store only a single read-only version of a given read/write fileset on any File Server machine. The **fts addsite** command prevents you from defining multiple replication sites for a fileset on the same File Server machine. Also, with Release Replication, you must choose the File Server machine on which the read/write fileset resides as the first replication site.

17.4.1.2.1 Adding a Replication Site

To add a replication site for a fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Select a replication site for the read/write fileset that is to be replicated. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate; the **fts lsft** command can be used to check the size of the read/write fileset.

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

3. Enter the **fts addsite** command to add the replication site. If Release Replication is being used, you must define the first replication site on the same File Server machine as the read/write fileset.

   $ **fts addsite -fileset** {*name* | *ID*} **-server** *machine* **-aggregate** *name* [**-maxsiteage** *interval*]

The **-maxsiteage** *interval* option can be used to override the DefaultSiteAge set with the **fts setrepinfo** command.

Repeat the **fts addsite** command for each replication site you want to define for the fileset.

17.4.1.2.2 Removing a Replication Site

To remove a replication site for a fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

   Note: If the fileset uses Release Replication and you are using the **fts rmsite** command to remove the replication site and replica on the same machine as the read/write fileset, you must also be included in the **admin.ft** file on the machine specified by the **-server** option. (The first replication site defined for a fileset that uses Release Replication must be on the same File Server machine as the read/write fileset.)

2. Enter the **fts rmsite** command to remove a replication site and to instruct the Replication Server at the site to remove the read-only replica of the fileset. If the fileset uses Release Replication and the replication site is on the same File Server machine as the read/write fileset, the **fts rmsite** command itself removes the replica.

   $ **fts rmsite -fileset** {*name* | *ID*} **-server** *machine* **-aggregate** *name*

   Repeat the **fts rmsite** command for each replication site and read-only fileset you want to remove for a fileset.

17.4.2 Creating Read-Only Filesets

The following subsections describe how to

- Use Release Replication to replicate a fileset with the **fts release** command
- Use Scheduled Replication to replicate a fileset
- Request that the Replication Server perform an immediate update of all replicas of a fileset or of only the replica at a given site with the **fts update** command

The operations described in the following subsections assume that the preparatory steps described in Section 17.4.1 have all been performed. Section 17.4.3 provides information about how to check the status of the Replication Server on a specific machine or the statuses of replicas of a fileset at one or all replication sites.

17.4.2.1 Using Release Replication to Create Read-Only Filesets

For a fileset that is using release replication, do the following to create or update the fileset's read-only replicas:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the read/write source fileset is stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the read/write source fileset is stored and for each machine on which a read-only replica is to reside. If necessary, issue the **bos lsadmin** command to verify the members of the list.
2. Use the **fts release** command to create or update the read-only replica of the read/write source fileset that resides on the same File Server machine as the source fileset. The Replication Servers at each replication site then place a copy of this read-only fileset at their respective sites.

   $ **fts release -fileset** {*name* | *ID*}

17.4.2.2 Using Scheduled Replication to Create Read-Only Filesets

Scheduled Replication takes place automatically, based on the parameters that were previously established with the **fts setrepinfo** command. You do *not* have to explicitly enter a command to initiate Scheduled Replication.

17.4.2.3 Requesting an Immediate Update of Read-Only Filesets

Issue the **fts update** command to request an immediate update of replicas at a specific or all replication sites:

$ **fts update -fileset** {*name* | *ID*} {**-all** | **-server** *machine*}

The **-fileset** *name* or *ID* option is the name or ID of the fileset whose replicas are to be updated.

The **-all** option specifies that all replicas of the fileset indicated with the **-fileset** option are to be updated. Use the **-all** option or use the **-server** option.

The **-server** *machine* option specifies the DCE pathname of a specific File Server machine whose replica of **-fileset** is to be updated. Use the **-server** option or use the **-all** option.

17.4.3 Displaying Replication Status

The operations described in the following subsections allow you to do the following:

- Display the status of the Replication Server (**repserver** process) on a File Server machine with the **fts statrepserver** command. Use this command to determine if the Replication Server at a replication site is functioning properly. If it is not, replicas stored on that machine may not be current.

- Display the statuses of all replicas of a fileset or the status of only the replica at a specific site with the **fts lsreplicas** command. Use this command to determine if the replicas of a fileset have been properly updated with the most recent version of the fileset. If replicas of a fileset are not current, use the **fts update** command to update them.

17.4.3.1 Displaying the Status of a Replication Server

Enter the **fts statrepserver** command to determine if the Replication Server on a File Server machine is running:

$ **fts statrepserver -server** *machine* [**-long**]

The **-long** option specifies that more detailed information about the Replication Server on **-server** is to be displayed. The additional information includes the status of each replica that is managed by the Replication Server.

17.4.3.2 Displaying the Statuses of Read-Only Filesets

Use the **fts lsreplicas** command to verify the statuses of a fileset's replicas at one or all of its replication sites:

$ **fts lsreplicas -fileset** {*name* | *ID*} {**-all** | **-server** *machine*}

The **-fileset** *name* or *ID* option is the name or ID of the fileset whose replicas are to be checked.

The **-all** option specifies that all replicas of the fileset indicated with the **-fileset** option are to be examined. Use the **-all** option or use the **-server** option.

The **-server** *machine* option specifies the DCE pathname of a specific File Server machine whose replica of the fileset indicated with the **-fileset** option is to be examined. Use the **-server** option or use the **-all** option.

17.5 Creating Backup DCE LFS Filesets

A backup fileset is a single copy of the read/write version of a DCE LFS fileset; backup versions of non-LFS filesets cannot be created. A backup fileset is stored at the same site as the read/write version and can serve as an online backup copy of a fileset. This process has the following purposes:

- It is the first step in using the Backup System to copy a DCE LFS fileset permanently to tape.
- It allows you to create backup copies without interrupting a user's work.
- It enables users to restore deleted or changed data themselves. A backup version captures the state of its read/write source at the time the backup is made. If you mount the backup version as a subdirectory of a user's home directory (with a suitable subdirectory name, such as **OldFiles** or **BackUp**), the user can restore files to the state they were in at the time of the backup without your help.

  If you do create and mount backup filesets for your users, you need to explain to them how often you will make the backups and remind them that the data in their backup filesets cannot be changed. They can, however, copy the data to a directory in a read/write fileset and use it there.

17.5.1 An Overview of Backup Filesets

The read-only version and backup version of a fileset are both copies of the read/write fileset, but they are created in different ways. When you create a read-only version at a site other than the read/write version, you make a copy of all the data in the source fileset. Each copy uses as much disk space as the source.

When you create a backup version of a fileset, the data is not copied. The system instead makes a clone of the array in the fileset header of the read/write fileset. The system then creates a fileset header for the backup version and places the array in it. The backup version resides at the same site (File Server machine and aggregate) as the read/write source. It has the same name as the read/write version, with the addition of a **.backup** extension.

The backup version preserves on disk any information that would be lost when changes are made to the read/write version. Because the array is a table of pointers between files in the fileset and their physical disk locations, it requires much less space than the data itself. A backup version requires additional space on a disk only if the read/write version changes; if the contents of the read/write version do not change much, the backup version requires little space.

These two different methods make backup and replication appropriate for different purposes. Because it points to specific disk blocks, the backup version can only be stored at the same site as its read/write source. Therefore, backing up a fileset does not make data available from multiple sites or reduce the load on a machine, as replication does. However, backing up is an efficient way to make previous versions of data such as user files available for restoration.

A backup version preserves the exact state of its read/write source at the time the backup was created. If you make a new version of each user's fileset every day at the same time, data that was deleted a week ago cannot be restored from the backup version because that backup version will have been overwritten six times. Similarly, if a user revises a file three times during the same day, there is no way to access the first and second revisions the next day; the backup version records only the third and final version, which is the version that existed in the read/write fileset when the backup was made.

Replication makes data available from multiple sites and can reduce the load on any particular machine. However, it also requires disk space because each read-only version is as large as the read/write version at the time the read-only version is created.

17.5.2 Backup Options

You can use the **fts clone** command to back up a single read/write DCE LFS fileset; you can use the **fts clonesys** command to back up multiple read/write DCE LFS filesets at one time. You can combine the options available with the **fts clonesys** command in different ways to indicate the

filesets that are to be backed up; the following list summarizes the possible combinations. To back up

- All filesets in the local cell, specify no options
- All filesets in the local cell with a name beginning with the same character string (for example, **sys.** or **user.**), specify the string with the **-prefix** option
- All filesets on a File Server machine, specify the machine's name with the **-server** option
- Filesets on a specific aggregate on a File Server machine, specify both the **-server** and **-aggregate** options
- Filesets with a certain prefix on a specific File Server machine, specify both the **-prefix** and **-server** options
- Filesets with a certain prefix on a specific aggregate on a File Server machine, specify the **-prefix**, **-server**, and **-aggregate** options

You may want to make backup versions of particular filesets every day at the same time. You can issue the **fts clonesys** or **fts clone** commands at the console, or you can create a **cron** entry in the **BosConfig** file on each File Server machine where you want to back up filesets.

The following example creates a **cron** process called **backupusers** in the **BosConfig** file on the machine named **fs3**. The process executes every day at 5:30 a.m., making a backup version of every fileset in the cell's filespace that has a name that starts with **user**. The **-localauth** option allows the process, which runs unauthenticated, to use the DFS server principal of **fs3** to execute the privileged **fts clonesys** command.

```
$ bos create /.../abc.com/hosts/fs3 backupusers cron "dcelocal/bin/fts
        clonesys -prefix user -localauth" 5:30
```

17.5.3 Creating and Mounting Backup Filesets

To create and mount backup filesets, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on each machine on which a backup fileset is to reside, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of a fileset to be backed up resides. If necessary, issue the **bos lsadmin** command to verify the members on an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for each directory in which a backup fileset is to be mounted. If necessary, issue the **acl_edit -l** command to check the permissions for the directories.
3. Issue the **fts clone** command to create a single backup version of a read/write source fileset. The backup version is placed at the same site as the read/write source, with the same name as the source fileset but with the addition of a **.backup** extension.

   $ **fts clone -fileset** {*name* | *ID*}

   Issue the **fts clonesys** command to create a backup version of every read/write fileset that shares the same prefix or site. Each backup version is placed at the same site as its read/write source, with the same name, and with a **.backup** extension.

   $ **fts clonesys** [**-prefix** *string*] [**-server** *machine*] [**-aggregate** *name*]

   The **-prefix** *string* option is the initial string in the name of each read/write fileset that you want to back up.
4. Enter the **fts crmount** command to create a mount point for each backup fileset. The contents of filesets are inaccessible until you perform this operation.

   $ **fts crmount -dir** *directory_name* **-fileset** *name*.**backup**

   The **-dir** *directory_name* option is the location for the root directory of the backup fileset; the specified location must not already exist. However, the parent directory of the mount point must exist in the

DCE namespace. Include a complete pathname unless you want to mount the fileset in the working directory.

The **-fileset** *name***.backup** option is the full name of the backup fileset, with the **.backup** extension.

Repeat the **fts crmount** command for each backup fileset created in the previous step. Note that you do not need to create a mount point for a backup fileset; for example, the DFS Backup System does not require a mount point to locate a backup fileset.

17.6 Using Mount Points

To make a DCE LFS or non-LFS fileset visible in the DCE namespace, you must use the **fts crmount** command to create a mount point for it in the DFS filespace. Mount points create the appearance of a single, seamless file system even though different filesets are stored on different File Server machines. A fileset's contents are visible and accessible as files and directories in the filespace only when the fileset is mounted at a directory location; they are not accessible if the fileset is not mounted. After the mount point is created, the fileset is automatically mounted; you never need to explicitly mount it again.

A mount point is a specific type of symbolic link stored in the file system. It acts as an association between a directory location and a fileset. During file retrieval, the Cache Manager traverses the file's complete pathname, continuing until it finds the file. When the Cache Manager encounters a mount point, it reads the mount point to learn which fileset is associated with that directory location and contacts the Fileset Location Server to locate the fileset in the cell. It then finds, as the root of the fileset, an actual directory structure and gives the directory the same name as the mount point. The fileset's root directory provides a pathname to all of the files, subdirectories, and mount points in the fileset. The Cache Manager checks the next element in the pathname in the root directory and interprets any further mount points it finds until it reaches the fileset that contains the requested file.

It is recommended that you *not* mount a fileset at more than one location in the file system. Creating multiple mount points can distort the hierarchical nature of the file system. The Cache Manager stores a single pointer to the

directory that contains the root directory of each fileset; the Cache Manager can become confused about which pathname to follow when searching for a file in a fileset with multiple mount points. This is true even if you specify the full pathname of a file. Create multiple mount points for a fileset sparingly (only in a very limited number of troubleshooting and testing situations). Remove the extraneous mount points as soon as they are no longer necessary.

A mount point can be characterized by its

- Fileset type (read/write, read-only, or backup)
- Mount point type (regular or read/write)
- Location type (cellular, local, or global root)

Section 17.6.1 describes the different types of mount points in greater detail. The **fts lsmount** command can be used to examine mount points to determine their types and the filesets they name. The command displays the following message for each directory that is a mount point:

```
'directory_name' is a mount point for fileset 'fileset_name'
```

In the output, *directory_name* is the name of a directory you specify, and *fileset_name* is the name of the fileset for which *directory_name* serves as a mount point. The command also provides the following information about the directory and fileset:

- A # (number sign) precedes the fileset name if the directory is a regular mount point.
- A % (percent sign) precedes the fileset name if the directory is a read/write mount point.
- A cellname and a : (colon) precede the fileset name if the directory is a cellular mount point; a # (number sign) or % (percent sign) precedes the cellname.
- An ! (exclamation point) replaces the fileset name if the directory is a global root mount point.

The **fts delmount** command is used to remove mount points. The fileset that is referenced by a removed mount point remains, but its contents are not accessible until another mount point is created.

17.6.1 Types of Mount Points

There are several different categories of mount points. A mount point can be distinguished by its fileset type, its mount point type, or its location type. The following subsections describe the different types of mount points.

17.6.1.1 Fileset Type

The first characteristic of a mount point determines which type of fileset is named in the mount point. A read-only or backup fileset has a **.readonly** or **.backup** extension; a read/write fileset has no extension.

When a mount point names a fileset with a **.readonly** or **.backup** extension, the Cache Manager uses only the specified fileset. The Cache Manager never accesses the read/write version of a fileset when it encounters a mount point that names a read-only or backup version; if the explicitly named version is unavailable, the Cache Manager reports an error. However, depending on the mount point type, the Cache Manager can access the read-only version of a fileset when it encounters a mount point that names the read/write version, which is the fileset name that has no extension.

17.6.1.2 Mount Point Type

The second characteristic determines the mount point type. The majority of mount points are *regular mount points*. When the Cache Manager encounters a regular mount point, it checks the version of the fileset that the mount point indicates—read/write, read-only, or backup. If the read-only or backup version is indicated, the Cache Manager uses that version. If the read/write version is indicated, the Cache Manager evaluates the type of fileset in which the mount point itself resides.

If the regular mount point for a read/write fileset resides in a read/write fileset, the Cache Manager attempts to access only the read/write version. If the read/write version does not exist or is inaccessible, the Cache Manager cannot access the fileset.

If the regular mount point for a read/write fileset resides in a read-only fileset, the Cache Manager attempts to access a read-only version of the

fileset first. If no read-only versions exist, the Cache Manager attempts to access the read/write version of the fileset. If one or more read-only versions exist but all are unavailable, the Cache Manager cannot access the fileset; it does not attempt to access the read/write version.

Regular mount points allow the Cache Manager to retrieve files better than other types of mount points because they allow the Cache Manager to access read-only filesets whenever possible. There are normally several different instances of a read-only fileset, but only one copy of the read/write version. Because more read-only copies are usually available, it is better to access the copies as often as possible.

A much less common type of mount point is a *read/write mount point*. Read/write mount points must name the read/write version of a fileset. When the Cache Manager encounters a read/write mount point, it attempts to access only the read/write version of the fileset, regardless of the type of fileset in which the mount point resides. If the read/write version does not exist or is inaccessible, the Cache Manager cannot access the fileset.

You usually mount read/write filesets with regular mount points. A regular mount point is explicitly *not* a read-only mount point. The Cache Manager can still access the read/write fileset when it encounters a regular mount point if no other versions exist or if it is already on a read/write path traversal.

The least common type of mount point is the *global root mount point*. This type of mount point is used to mount the root of the DCE global namespace. It is typically used only in filesets that serve as top-level directories for diskless machines.

17.6.1.3 Location Type

The third characteristic of a mount point determines the location type of the mount point. A mount point that indicates to the Cache Manager that the fileset resides in a different cell and identifies which cell is referred to as a *cellular mount point*. If a mount point is not cellular, the Cache Manager assumes that the fileset resides in the same cell as the mount point.

A cellular mount point can be either regular or read/write. A mount point that is both regular and cellular and that names a read-only or backup fileset directs the Cache Manager to the other cell to access the indicated version of the fileset. However, a mount point that is both regular and cellular and

that names a read/write fileset directs the Cache Manager to the other cell to access a read-only version of the fileset. If possible, the Cache Manager traverses a read-only path in the other cell. This is true even if the cellular mount point resides in a read/write fileset.

A mount point that is both read/write and cellular directs the Cache Manager to cross to the named cell and access the read/write version. This type of mount point is not recommended. Accessing a read/write fileset causes the File Exporter on the machine where the fileset resides to maintain tokens, imposing an extra load on the system in the other cell.

17.6.2 Manipulating Mount Points

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

17.6.2.1 Creating a Mount Point

To create a mount point, do the following:

1. Verify that you have the necessary permissions for the directory in which the fileset is to be mounted. If the directory resides in a DCE LFS fileset, you must have **w** (write), **x** (execute), **c** control, and **i** (insert) ACL permissions for the directory; if necessary, issue the **acl_edit -l** command to list the permissions for the directory. If the directory resides in a non-LFS fileset, you must have **w** and **x** permissions for the directory.
2. Enter the **fts crmount** command to create a mount point for a fileset:

   $ **fts crmount -dir** *directory_name* {**-fileset** {*name* | *ID*} | **-global**}
   [**-cell** *cellname*] [**-rw**] [**-fast**]

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

The **-global** option indicates that the mount point is for the root of the DCE global namespace. Use this option with diskless machines to create a mount point for the root of the DCE namespace under the top-level diskless directory. Use the **-global** option or use the **-fileset** option.

The **-cell** *cellname* option specifies the location type of the mount point as cellular. Use this option to specify the name of the cell in which the fileset resides. Omit this option to create a mount point for a fileset in the cell in which the mount point's parent directory resides.

The **-rw** option specifies the type of the mount point as read/write. The Cache Manager accesses only the read/write version of the fileset. If the **-rw** option is used, the **-fileset** option must name the read/write version. Omit the **-rw** option to create a regular mount point.

The **-fast** option specifies that the existence of the fileset indicated with the **-fileset** option and the cell indicated with the **-cell** option, if specified, are not to be verified. By default, **fts** verifies the existence of both, displaying a warning if either does not exist; it creates the mount point regardless of whether they exist.

The following examples illustrate the creation of a mount point for a user fileset. Initially, only the user filesets named **user.pat** and **user.terry** are mounted at **/.../abc.com/fs/usr**.

```
$ cd /.../abc.com/fs/usr
$ ls

pat        terry
```

The **fts crmount** command is then used to mount the fileset named **user.vijay** at **/.../abc.com/fs/usr**. Because the **-rw** option is omitted, the fileset is mounted with a regular mount point.

```
$ fts crmount /.../abc.com/fs/usr/vijay user.vijay
$ ls

pat       terry     vijay
```

17.6.2.2 Listing a Mount Point

To list a mount point, do the following:

1. Verify that you have the **r** (read) permission for each mount point that you want to examine. If necessary, issue the **acl_edit -l** command to list the permissions for a mount point that resides in a DCE LFS fileset.
2. Enter the **fts lsmount** command to list information about one or more mount points:

   ```
   $ fts lsmount -dir directory_name...
   ```

   The **-dir** *directory_name* option specifies the name of each mount point about which you want to list information.

   The following example lists the mount point for the fileset **user.vijay**, which was created in the previous example:

   ```
   $ fts lsmount vijay

   'vijay' is a mount point for fileset '#user.vijay'
   ```

17.6.2.3 Removing a Mount Point

To remove a mount point, do the following:

1. Verify that you have the necessary permissions for each directory from which you want to remove a mount point. If a directory resides in a DCE LFS fileset, you must have **w** (write), **x** (execute), and **d** (delete) ACL permissions for the directory; if necessary, issue the **acl_edit -l** command to list the permissions for the directory. If a directory resides in a non-LFS fileset, you must have **w** and **x** permissions for the directory.
2. Enter the **fts delmount** command to remove one or more mount points:

   $ fts delmount -dir *directory_name...*

   The **-dir** *directory_name* option specifies the name of each mount point that you want to remove.

   The following example removes the mount point for the fileset **user.vijay**. The fileset itself is not deleted, just its mount point. As a result, the fileset is no longer visible in the DCE namespace.

   ```
   $ ls

   pat       terry     vijay

   $ fts delmount vijay
   $ ls

   pat       terry
   ```

Chapter 18

Managing Filesets

This chapter explains in detail how to manage filesets in DFS. It describes how to obtain information about filesets, aggregates, and partitions, how to set fileset quotas, and how to rename, move, dump, restore, and delete filesets. It also details how to verify and maintain the consistency of your filesets and how to recover from possible file system problems. Where applicable, the management of filesets in other file systems is discussed.

Chapter 17 provides details about how to export aggregates and partitions, and how to create, replicate, backup, and mount filesets. It also provides a general overview of filesets and the differences between DCE LFS and non-LFS filesets. (See Chapter 17 for introductory information about filesets.)

18.1 An Overview of Fileset Terminology

DCE LFS aggregates are similar to the disk partitions that are found in the UNIX and other operating systems. However, a DCE LFS aggregate also supports specialized fileset-level operations, such as quota-checking and cloning, and low-level operations, such as logging of metadata.

Each DCE LFS aggregate exported to the DCE namespace can house multiple filesets; filesets stored on DCE LFS aggregates are referred to as DCE LFS filesets. Each exported non-LFS disk partition can house only a single fileset; file systems stored on non-LFS partitions are referred to as non-LFS filesets. Most of the specialized features supported for DCE LFS filesets are not supported for non-LFS filesets.

In DFS, only a single type of each non-LFS fileset is available: the version that was made available when the partition on which it resides was exported. However, three types of each DCE LFS fileset are available:

- A *read/write version* of a DCE LFS fileset contains modifiable versions of the files and directories in that fileset. Every DCE LFS fileset begins as a single read/write fileset. Other fileset types are derived from the read/write version by creating an exact copy, or replica, of all of the data in the read/write, source fileset. There can be only one read/write version of a fileset.
- A *read-only fileset* is a copy of a read/write, source DCE LFS fileset. Read/write filesets can be replicated and placed at various sites in the file system. Every read-only copy of a fileset shares the name of the source fileset, with the addition of a **.readonly** extension. If a read/write source changes, its read-only replicas can be updated to match. Every read-only copy of a read/write fileset is generally the same; however, replicas at different sites can differ in controlled ways according to the replication parameters associated with the fileset.
- A *backup fileset* is a clone of a read/write, source DCE LFS fileset. It is stored at the same site and with the same name as the source, with the addition of a **.backup** extension. A backup fileset contains a copy of the array that maps files to their disk locations. It maintains pointers to the data that existed in the source fileset when the backup was made.

For both DCE LFS and non-LFS filesets, the Fileset Location Server (FL Server) maintains the Fileset Location Database (FLDB). The database records information about the location of all filesets in a cell. Every

read/write fileset has an entry in the FLDB. Each entry for a DCE LFS fileset also includes information about the fileset's read-only and backup versions. For each fileset, the information in the entry includes fileset names, ID numbers, site definitions, and status flags for the sites.

Information about DCE LFS filesets is also stored in fileset headers at each site that contains a copy of the fileset. Fileset headers are part of the data structure that records the disk addresses on the aggregate of the files in the fileset. Fileset headers also record the fileset's name, ID number, size, status flags, and the ID numbers of its copies. Because the header records some of the same information that appears in the FLDB, the **fts** program can access the information in the header if the FLDB becomes unavailable. The FLDB entry and the fileset header must always be synchronized; any **fts** commands that affect fileset status automatically record the change in the appropriate FLDB entry.

(See Chapter 17 for detailed information about DCE LFS filesets, non-LFS filesets, and how the two types of filesets are created.)

18.2 Standard Options and Arguments

When using the **fts** commands to create, manipulate, or delete filesets, you can often add the **-verbose** option to receive detailed information from the **fts** program about its actions as it executes the command. This can be particularly helpful if an operation fails for a reason that you do not understand. The amount of additional information produced by the **-verbose** option varies for different commands.

The following options and arguments are common to many of the commands described in this chapter. If an option or argument is not described with a command in the text, a description of it appears here. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete details about each command.)

- The **-fileset** *name* option is the complete name (for example, **user.sandy**) or ID number (for example, **0,,34692**) of the fileset to be used in the command.
- The **-server** *machine* option is the File Server machine to be used for the command. Use the DCE pathname (for example, **/.../abc.com/hosts/fs1**) in all commands unless otherwise indicated.

- The **-aggregate** *name* option is the device name (for example, **/dev/lv01**), the aggregate name (for example, **lfs1** or **/usr**), or the aggregate ID (for example, **3** or **12**) of the aggregate or partition to be used in the command. These identifiers are specified in the first, second, and fourth fields of the entry for the aggregate or partition in the *dcelocal*/**var/dfs/dfstab** file.
- The **-cell** *cellname* option specifies the cell with respect to which the command is to be run (for example, **abc.com**). The default is the local cell of the issuer of the command.
- The **-noauth** option directs the **fts** program to use the unprivileged identity **nobody** as the identity of the issuer of the command. If DFS authorization checking has been disabled with the **bos setauth** command, the identity **nobody** has the necessary privileges to perform any operation. (See Chapter 15 for information about disabling DFS authorization checking.) If you use the **-noauth** option, do not use the **-localauth** option.
- The **-localauth** option directs the **fts** program to use the DFS server principal of the machine on which the command is issued as the identity of the issuer. Each DFS server machine has a DFS server principal stored in the Registry Database. A DFS server principal is a unique, fully qualified principal name that ends with the string **dfs-server** (for example, **/.../abc.com/hosts/fs1/dfs-server**). Do not confuse a machine's DFS server principal with its unique **self** identity. (See Chapter 17 for information about DFS server principals.)

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

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

18.3 Listing Fileset Information

The following commands are available for listing information about filesets:

- The **fts lsfldb** command is used to list information about filesets from the FLDB.
- The **fts lsheader** command is used to list information from fileset headers on File Server machines and aggregates.
- The **fts lsft** command is used to list information from both the FLDB entry and the fileset header of a specific fileset.

If you know only the name of a file or directory that is housed in a fileset, you can use the **fts lsquota**, **fts lsft**, and **cm whereis** commands to access information about the fileset. (See Section 18.3.4 for complete instructions on using these commands with file or directory names.)

18.3.1 Listing FLDB Information

The **fts lsfldb** command is used to display information from entries in the FLDB for both DCE LFS and non-LFS filesets. By combining the command's options in different ways, you can tailor the type of information to be displayed, as follows:

- To display every FLDB entry, do not supply any options.
- To display every FLDB entry that mentions a specific File Server machine as the site of any version of a fileset, specify the machine name with the **-server** option.
- To display every FLDB entry that mentions a specific aggregate on a specific File Server machine as the site of any version of a fileset, specify both the **-server** and **-aggregate** options.
- To display the FLDB entries for filesets with locked entries, use the **-locked** option alone or with the **-server** option (and optionally the **-aggregate** option).
- To display a single FLDB entry, specify the fileset name or ID number with the **-fileset** option.

To list information about fileset entries in the FLDB, enter the **fts lsfldb** command with the options described previously:

```
$ fts lsfldb [-fileset {name | ID}] [-server machine] [-aggregate name] [-locked]
```

The **-locked** option lists information only for filesets with locked FLDB entries. Use the **-locked** option alone or with the **-server** option (and optionally the **-aggregate** option), or use the **-fileset** option to view the FLDB entry for a specific fileset.

Interpreting the Output

The output appears in the following order:

- The fileset's name.
- The fileset IDs of the read/write, read-only, and backup versions of the fileset.
- For each version of a fileset, a status flag of `valid` or `invalid` indicates whether it actually exists at some site. For the read-only version, it indicates whether a replication site is defined.
- The number of sites at which a version of the fileset exists.
- An indicator if the FLDB entry is locked. The indicator is omitted if the entry is not locked.
- The replication parameters that are associated with the fileset.
- Information identifying the File Server machines and aggregates (sites) where read/write (`RW`), read-only (`RO`), or backup (`BK`) versions of the fileset reside.
- For a read-only version, the MaxSiteAge defined for that site. For a read/write version, `0:00:00` is displayed because no MaxSiteAge is associated with that version.
- The abbreviated DCE principal name of each File Server machine on which a version of the fileset resides, and the name of the group that owns the server entry for the machine or `<nil>` if no group owns the server entry.

If the output includes more than one FLDB entry, information about the filesets is displayed in alphabetical order by fileset name. The last line of the output displays the total number of entries that were successfully reported and the total number of entries that were not reported (the number of entries that failed).

Following is an example of the output that this command generates for a single DCE LFS fileset:

```
$ fts lsfldb user.terry
```

```
user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
number of sites: 1
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00;
  minRepDelay=0:05:00; defaultSiteAge=0:30:00
   server         flags    aggr    siteAge principal  owner
fs3.abc.com       RW,BK    lfs1    0:00:00 hosts/fs3  <nil>
```

18.3.2 Listing Fileset Header Information

The **fts lsheader** command displays the fileset header from every DCE LFS fileset on a File Server machine or one of its aggregates. You control the amount of information to be displayed by including the **-fast** option to display only the fileset ID number of each fileset, including the **-long** option to display all available information about each fileset, or omitting both options to display a single line of information about each fileset. Information about the filesets is displayed in numeric order by fileset ID number if the **-fast** option is used; otherwise, it is displayed in alphabetical order by fileset name. To view the header of a single DCE LFS fileset, use the **fts lsft** command, as described in Section 18.3.3.

Non-LFS filesets do not have DCE LFS fileset headers. However, the **fts lsheader** command can still be used to display some local information, such as fileset ID numbers, that is stored in the *dcelocal*/**var/dfs/dfstab** file on a File Server machine.

To list information from fileset headers, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the filesets reside. If necessary, issue the **bos lsadmin** command to verify the members of the list.
2. Issue the **fts lsheader** command to display information from fileset headers. Include the **-fast** or **-long** option to see less or more information.

   $ **fts lsheader -server** *machine* [**-aggregate** *name*] [{**-fast** | **-long**}]

Interpreting the Output

Following is an example of the output from this command when it is executed with the **-fast** option:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1 -fast

0,,196953
0,,196956
    .
    .
    .
0,,199845
0,,199846
```

When you omit both the **-fast** and **-long** options, the command produces the following information:

- The File Server machine name, aggregate name, and aggregate ID number where the filesets reside
- The total number of filesets on the aggregate
- Each fileset's name, with a **.readonly** or **.backup** extension, if appropriate
- Its fileset ID number
- Its type (RW for read/write, RO for read-only, or BK for backup)
- Its size in kilobytes

- Its status (`On-line`, `Off-line`, or an error indicator)
- The total number of filesets that are online, the total number of filesets that are offline, and the total number of filesets that are busy

Following is an example of the output from the command when it is run with no options:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1

Total filesets on server fs3 aggregate lfs1 (ID 10): 16
user.terry              0,,196953 RW     5071 K  On-line
user.wvh                0,,196956 RW     4955 K  On-line
    .
    .
    .
Total filesets on-line 15; total off-line 1;
total busy 0
```

When you include the **-long** option, the command displays the following additional information for each fileset:

- Whether it is a DCE LFS (`LFS`) or non-LFS fileset
- Additional internal information about the fileset and its access status
- The ID numbers of the parent, clone, and backup filesets that are related to the fileset
- The ID numbers of the low-level backing and low-level forward filesets that are related to the fileset
- The version number of the fileset
- The maximum quota of the read/write version of the fileset, in kilobytes
- The day, date, and time when the fileset was created (replicated or cloned for a read-only or backup fileset)
- The day, date, and time when the contents of the fileset were last updated (this is the same as the creation time for a read-only or backup fileset)

Following is an example of the output from the command when it is executed with the **-long** option:

```
$ fts lsheader /.../abc.com/hosts/fs3 /dev/lfs1 -long

Total filesets on server fs3 aggregate lfs1 (ID 10): 16
user.terry 0,,196953 RW LFS  5071 K  states 0x4009 accStatus 0x0  On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    MaxQuota    15000 K
    Creation Tue Oct 15 16:45:16 1991
    Last Update Fri Nov 22 11:36:00 1991

user.wvh 0,,196956 RW LFS  4955 K  states 0x9 accStatus 0x0  On-line
    .
    .
    .
Total filesets on-line 15; total off-line 1;
total busy 0
```

18.3.3 Listing FLDB and Fileset Header Information

You can use the **fts lsft** command to display information from both the FLDB and the fileset header for a single fileset. The output from the **fts lsft** command consists of the output from the **fts lsheader** command with the **-long** option followed by the output from the **fts lsfldb** command. You can indicate the fileset about which information is to be displayed in one of two ways: by specifying the name of a file or directory that is stored in the fileset with the **-path** option, or by specifying the name or fileset ID number of the fileset with the **-fileset** option.

Because the **fts lsft** command retrieves information from the fileset header, you can examine the read-only or backup version of a DCE LFS fileset by adding the **.readonly** or **.backup** extension to the name of the fileset specified with the **-fileset** option or by specifying the ID number of the read-only or backup version. An error message is displayed if the read/write version no longer exists and you fail to specify the **.readonly** or **.backup** extension with the name of the fileset.

If multiple read-only replicas of a DCE LFS fileset exist, you can use the **-server** option to indicate the name of the File Server machine that houses the specific replica to be examined. While all replicas of a fileset are identical by default, indicating a specific replica can be useful if network or hardware problems caused only some of a fileset's replicas to be updated. Omit the **-server** option to display information about the replica at the fileset's oldest read-only site. The **-server** option is always unnecessary when the command is used to examine the read/write or backup version of a fileset.

A read-only version of a DCE LFS fileset can exist independently of a read/write version if the **fts rmsite** command is not used to remove its site when the **fts delete** command is used to remove the read/write version. A backup version of a DCE LFS fileset can exist independently of a read/write version if the **fts delete** operation used to remove the read/write version is interrupted before completion; for example, if a system or hardware failure stops a delete operation after the read/write version is removed but before the backup version is removed. (See Section 18.10 for more information about deleting DCE LFS—and non-LFS—filesets.)

Because non-LFS filesets do not have fileset headers, the **fts lsft** command displays much less fileset header information for non-LFS filesets than it does for DCE LFS filesets.

To display information about a fileset from both its FLDB entry and its fileset header, enter the **fts lsft** command:

```
$ fts lsft [{-path {filename | directory_name} | -fileset {name | ID}}]
      [-server machine]
```

The **-path** *filename* or *directory_name* option is the name of a file or directory in the fileset. Use the **-path** option or use the **-fileset** option to specify the name or ID number of the fileset; omit both options to display information about the fileset that houses the working directory.

The **-server** *machine* option names the File Server machine that houses the version of the fileset about which information is to be displayed. This option is useful for examining a particular read-only replica of a DCE LFS fileset for which multiple replicas exist.

Following is an example of the output for this command. The fileset ID number is used to indicate the fileset about which information is to be displayed; the leading 0 (zero) and commas are omitted from the ID number.

```
$ fts lsft -fileset 196953

user.terry 0,,196953 RW LFS  5071 K  states 0x4009 accStatus 0x0  On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    Parent 0,,196953   Clone 0,,0   Backup 0,,196955
    llBack 0,,0        llFwd 0,,0   Version 0,,25963
    MaxQuota    15000 K
    Creation Tue Oct 15 16:45:16 1991
    Last Update Fri Nov 22 11:36:00 1991

user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
number of sites: 1
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00;
  minRepDelay=0:05:00; defaultSiteAge=0:30:00
   server        flags   aggr   siteAge principal  owner
fs3.abc.com      RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

18.3.4 Determining Other Fileset Information

The following subsections describe commands that are used to obtain information about a fileset when you know only its name, its ID number, or the name of a file or directory that it contains. These subsections include descriptions of the following commands:

- The **fts lsquota** command, which is used to determine the name of a fileset from the name of a file or directory that it houses. Note that the primary usage of the **fts lsquota** command is to list fileset quota, as described in Section 18.6, not to determine fileset names.
- The **fts lsft** command, which is used to determine the ID number of a fileset from its name or from the name of a file or directory that it houses.
- The **fts lsfldb** and **cm whereis** commands, which are used to learn the location of a fileset from its name or ID number or from the name of a file or directory that it houses.

The **cm whereis** command is from the DFS **cm** command suite. (See Chapter 19 for more information about using **cm** commands; see the DCE DFS portion of the *OSF DCE User's Guide and Reference* for more detailed information about the **cm whereis** command.)

18.3.4.1 Learning Fileset Names

To determine fileset names from file or directory names, enter the **fts lsquota** command with the **-path** option:

$ **fts lsquota** [**-path** {*filename* | *directory_name*}...]

The **-path** *filename* or *directory_name* option is the name of a file or directory in each unknown fileset. You can include multiple files or directories from different filesets. Omit this option to learn the name of the fileset that houses the working directory.

Following is an example of the output from this command:

$ **fts lsquota /.../abc.com/fs/usr/terry**

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

18.3.4.2 Learning Fileset ID Numbers

To learn a fileset ID number from a file or directory name, enter the **fts lsft** command with the **-path** option:

$ **fts lsft** [**-path** {*filename* | *directory_name*}]

The **-path** *filename* or *directory_name* option is the name of a file or directory in an unknown fileset. Omit this option to learn the ID number of the fileset that houses the working directory.

The following example displays the fileset ID number of the fileset that contains the current working directory:

$ fts lsft

```
 user.terry 0,,196953 RW LFS  5071 K  states 0x4009 accStatus 0x0  On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    .
    .
    .
   server         flags    aggr    siteAge principal  owner
fs3.abc.com       RW       lfs1    0:00:00 hosts/fs3  <nil>
```

To learn a fileset ID number from a fileset name, enter the **fts lsft** command with the **-fileset** option:

$ fts lsft -fileset {*name* | *ID*}

The following example displays the fileset ID number of the fileset whose name is **user.terry**:

$ fts lsft -fileset user.terry

```
 user.terry 0,,196953 RW LFS  5071 K  states 0x4009 accStatus 0x0  On-line
    fs3.abc.com, aggregate lfs1 (ID 10)
    .
    .
    .
   server         flags    aggr    siteAge principal  owner
fs3.abc.com       RW       lfs1    0:00:00 hosts/fs3  <nil>
```

18.3.4.3 Learning Fileset Locations

To learn the location of a fileset from the fileset name or ID number, enter the **fts lsfldb** command with the **-fileset** option:

$ fts lsfldb -fileset {*name* | *ID*}

Following is an example of this command and its output:

$ fts lsfldb user.terry

```
user.terry
        readWriteID  0,,196953  valid
        readOnlyID   0,,196594  invalid
        backupID     0,,196595  valid
number of sites: 1
  Sched repl: maxAge=2:00:00; failAge=1d0:00:00; reclaimWait=18:00:00;
  minRepDelay=0:05:00; defaultSiteAge=0:30:00
   server         flags   aggr   siteAge principal  owner
fs3.abc.com       RW,BK   lfs1   0:00:00 hosts/fs3  <nil>
```

To learn the locations of filesets from file or directory names, enter the **cm whereis** command:

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

The **-path** *filename* or *directory_name* option is the name of a file or directory in an unknown fileset. You can include multiple files or directories from different filesets. Omit this option to learn the location of the working directory.

Following is an example of the output from this command:

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

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

18.4 Listing Aggregate and Partition Information

The following commands are available for listing information about any aggregate or partition (non-LFS aggregate) that is exported to the DCE namespace:

- The **fts lsaggr** command is used to list all exported aggregates or partitions on a File Server machine.
- The **fts aggrinfo** command is used to list information about the amount of disk space available on a specific aggregate or partition or on all exported aggregates and partitions on a File Server machine.

These commands are especially useful when exporting additional aggregates or partitions from a machine, when creating read/write or read-only filesets on an aggregate, or when moving filesets between machines.

18.4.1 Listing Aggregates and Partitions

The **fts lsaggr** command is used to list the following information about each exported aggregate or partition on a File Server machine. The information is specified for each aggregate and partition in the *dcelocal*/**var/dfs/dfstab** file.

- The aggregate name, which is specified in the second field of the **dfstab** file
- The device name, which is specified in the first field of the **dfstab** file
- The aggregate ID, which is specified in the fourth field of the **dfstab** file
- The file system type, which is specified in the third field of the **dfstab** file

Note: You can issue the **dfsexport** command with no options on a File Server machine to list the aggregates and partitions with entries in the *dcelocal*/**var/dfs/dfsatab** file on the machine. This approach also displays an accurate listing of all exported aggregates and partitions. The **dfsexport** command copies the entry for an aggregate or a partition from the **dfstab** file to the **dfsatab** file when it exports it. Similarly, it removes the entry for an aggregate or a partition from the **dfsatab** file when it detaches it.

Enter the **fts lsaggr** command to list information about all of the exported aggregates and partitions on a File Server machine:

$ **fts lsaggr -server** *machine*

The following example shows that two non-LFS partitions and two DCE LFS aggregates are exported from the File Server machine named **fs1**:

$ **fts lsaggr /.../abc.com/hosts/fs1**

```
There are 4 aggregates on the server /.../abc.com/hosts/fs1 (fs1.abc.com):
        /usr (/dev/lv02): id=3      (non-LFS)
        /tmp (/dev/lv03): id=4      (non-LFS)
        lfs1 (/dev/lfs1): id=10      (LFS)
        lfs2 (/dev/lfs2): id=11      (LFS)
```

18.4.2 Listing Disk Space on Aggregates and Partitions

The **fts aggrinfo** command lists information about the total amount of disk space and the amount of disk space that is currently available on exported aggregates and partitions. It can be used to obtain size information about a specific aggregate or partition or about all of the exported aggregates and partitions on a File Server machine. The command displays the following information about each aggregate and partition:

- The file system type (DCE LFS or non-LFS).
- The aggregate name.
- The device name.
- The number of kilobytes of disk space that is currently available on the aggregate or partition.
- The total number of kilobytes on the aggregate or partition.
- The number of kilobytes, if any, of disk space that is reserved for overdraw. Overdraw is additional disk space that is reserved for the partition by some non-LFS implementations in case the allocated size of the partition is exceeded. DCE LFS aggregates do not reserve overdraw disk space.

The **df** command available in the UNIX operating system displays roughly the same information as the **fts aggrinfo** command for non-LFS partitions, exported DCE LFS aggregates, and locally mounted DCE LFS filesets.

Enter the **fts aggrinfo** command to display information about the disk space that is available on a specific aggregate or partition or on all aggregates and partitions on a File Server machine:

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

The following example displays information about the disk space that is available on the aggregates and partitions that are exported from **fs1**:

$ **fts aggrinfo /.../abc.com/hosts/fs1**

```
Non-LFS aggregate /usr (/dev/lv02): 24048 K free out of total 98304
     (10923 reserved)
Non-LFS aggregate /tmp (/dev/lv03): 11668 K free out of total 12288
     (1365 reserved)
LFS aggregate lfs1 (/dev/lfs1): 100537 K free out of total 102400
LFS aggregate lfs2 (/dev/lfs2): 79957 K free out of total 81920
```

18.5 Increasing the Size of a DCE LFS Aggregate

DCE LFS aggregates are created with the **newaggr** command. The initial size of an aggregate is specified with the command's **-aggrsize** option. (See Chapter 17 for more information about the **newaggr** command.)

The **growaggr** command is used to increase the size of an existing DCE LFS aggregate. On operating systems that support logical volumes, the **growaggr** command is useful for increasing the size of an aggregate when the size of the logical volume on which it resides is increased. It can also be used to increase the size of an aggregate that was deliberately created smaller than the partition or logical volume on which it resides.

The size of the aggregate can be made as large as the size of the partition on which it resides. To determine the total number of 1024-byte blocks on the partition on which an aggregate resides without changing the size of the aggregate, specify the **growaggr** command's **-noaction** option and omit its **-aggrsize** option. To increase the size of an aggregate to the total size of the partition on which it resides, omit both the **-aggrsize** and **-noaction** options.

The size of an existing aggregate cannot be decreased. A size specified with the command's **-aggrsize** option must be at least three DCE LFS blocks greater than the current size of the aggregate. (The number of bytes in a DCE LFS block is defined on a per-aggregate basis with the **-blocksize** option of the **newaggr** command when an aggregate is created.) Specify both the **-aggrsize** and **-noaction** options with the command to determine whether the size specified with the **-aggrsize** option is valid without changing the present size of the aggregate.

To increase the size of a DCE LFS aggregate, do the following:

1. Log in as **root** on the machine on which the aggregate that is to be enlarged resides.
2. Issue the **growaggr** command to increase the size of the aggregate. (See the DCE DFS portion of the *OSF DCE Administration Reference* for more detailed information about the **growaggr** command.)

   # growaggr -aggregate *name* [**-aggrsize** *blocks*] [**-noaction**]

   The **-aggregate** *name* option is the device name or aggregate name of the DCE LFS aggregate to be grown. These identifiers are specified in the first and second fields of the entry for the aggregate in the */dce/local/***var/dfs/dfstab** file.

   The **-aggrsize** *blocks* option is the total number of 1024-byte blocks to be available on the specified aggregate. The number of 1024-byte blocks specified with this option cannot exceed the total size of the disk partition on which the aggregate resides, and it must be at least three DCE LFS blocks larger than the current size of the aggregate. Omit both the **-aggrsize** option and the **-noaction** option to increase the size of the aggregate to the total size of the disk partition on which it resides. Specify both the **-aggrsize** option and the **-noaction** option to determine whether the size specified with this option is valid without changing the current size of the aggregate.

   The **-noaction** option directs the command to display the total number of 1024-byte blocks on the disk partition on which the specified aggregate resides, provided the **-aggrsize** option is not specified. If the **-aggrsize** option is specified with the **-noaction** option, the command determines whether the specified size is valid. The current size of the aggregate is not affected if the **-noaction** option is used.

18.6 Setting and Listing Fileset Quota

By default, every newly created DCE LFS fileset has a maximum quota of 5000 kilobytes (1024-byte units). The **fts setquota** command can be used to modify the quota of a DCE LFS fileset. The **fts lsquota** command can be used to examine the quota that is available and used on a DCE LFS fileset.

Because it does not represent the amount of physical data stored on the fileset, the quota of a DCE LFS fileset can be larger than the size of the aggregate on which the fileset resides. Similarly, the combined quotas of all filesets on a DCE LFS aggregate can exceed the size of the aggregate. Assuming that all users who own filesets on an aggregate do not use all of their quota, you can allocate more quota than an aggregate actually contains to minimize user requests for additional quota. Also, if additional quota is allocated to filesets that reside on an aggregate whose size can be increased, the aggregate can be grown to accommodate the additional quota if necessary (see Section 18.5).

The size of a non-LFS fileset is equivalent to the size of the partition on which it is stored. In the UNIX operating system, you can use the **df** command to determine the size of a non-LFS partition. The **df** command can also be used to check the size of an exported DCE LFS aggregate, but it cannot be used to display the size of a DCE LFS fileset, unless the fileset is mounted locally. In addition, although you can use the **fts lsquota** command in DFS to check the space that is used and available on a non-LFS fileset, you cannot use the **fts setquota** command to set the quota of a non-LFS fileset.

18.6.1 Setting Quota for a DCE LFS Fileset

To set the quota for a DCE LFS fileset, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of the list.
2. Issue the **fts setquota** command:

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

The **-path** *filename* or *directory_name* option is the name of a file or directory on the fileset whose quota you want to set. Use the **-path** option or use the **-fileset** option to specify the name or ID number of the fileset whose quota you want to set.

The **-size** *kbytes* option is the maximum amount of disk space the fileset can occupy. Specify the value in kilobytes; a value of 1024 kilobytes equals 1 megabyte.

18.6.2 Listing Quota, Size, and Other Information for a Fileset

To display quota information about a fileset, enter the **fts lsquota** command:

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

The **-path** *filename* or *directory_name* option is the name of a file or directory on each fileset whose quota you want to display. You can include multiple files or directories from different filesets. Use the **-path** option or use the **-fileset** option to specify the name or ID number of each fileset whose quota you want to display. Omit both options to display quota information about the fileset that contains the working directory.

Following is an example of this command and its output. The << and <<WARNING messages indicate that the fileset named **user.jean** is over 90% full; the same message appears for an aggregate that is over 97% full. Ignore the values in the fileset information columns for the non-LFS fileset, **user.jlw**; the quota and usage information for a non-LFS fileset is equal to the same information displayed for the partition on which the fileset resides.

$ **fts lsquota /.../abc.com/fs/usr/terry /.../abc.com/fs/usr/jean /.../abc.com /fs/usr/jlw**

```
Fileset Name     Quota   Used   % Used   Aggregate
user.terry       15000   5071      34%     86% = 84538/98300 (LFS)
user.jean         5000   4955      99%<<   92% = 87436/98300 (LFS)
   <<WARNING
user.jlw             0      0       0%     84% = 8448/10000 (non-LFS)
```

18.7 Renaming Filesets

You can use the **fts rename** command to change the name of the read/write version of any DCE LFS or non-LFS fileset. When you change the name of a fileset's read/write version, the names of the read-only and backup versions of the fileset are automatically changed accordingly. When you change the name of a fileset, you must also replace any mount points that reference versions of the fileset by the old name with mount points that indicate the new name.

To rename a fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the read/write fileset resides, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), **i** (insert), and **d** (delete) ACL permissions for the directory in which you will replace the mount point. If necessary, issue the **acl_edit -l** command to list the permissions for the directory.
3. Issue the **fts rename** command to rename the fileset:

   $ **fts rename -oldname** *oldname* **-newname** *newname*

   The **-oldname** *oldname* option is the current name of the read/write fileset.

   The **-newname** *newname* option is the new name of the fileset. The new name can be no longer than 102 characters. (See Chapter 17 for more information on fileset naming conventions.)
4. Issue the **fts delmount** command to remove the mount point that indicates the fileset's old name:

   $ **fts delmount -dir** *directory_name...*

   The **-dir** *directory_name* option is the name of each mount point you want to remove.

5. Issue the **fts crmount** command to create a new mount point that indicates the fileset's new name:

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

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

   The **-fileset** *name* or *ID* option is the name or ID number of the fileset to be mounted. Use the name you specified with the **fts rename** command in the earlier step.

18.8 Moving DCE LFS Filesets

The **fts move** command allows you to move read/write versions of DCE LFS filesets between aggregates on the same machine or between machines. Filesets can be moved between sites in the same cell only. Non-LFS filesets cannot be moved.

Read/write filesets are the only types of filesets that you can move. When you move the read/write version of a fileset, the backup version is automatically deleted from the read/write site; you cannot move the backup version of a fileset. Use the **fts clone** command to create a backup fileset at the new site. All read-only versions of a read/write fileset remain unaffected when the read/write source moves; use the **fts rmsite** and **fts addsite** commands to remove one replication site and add another. You do not need to change the mount point for a fileset when you move it.

Move filesets to another machine if their current machine or disk must be removed for repair. Consider moving filesets if an aggregate becomes full or if a File Server machine becomes overloaded.

Note: You cannot move a DCE LFS fileset that is also mounted locally (as a file system on its File Server machine) to a different File Server machine; you can move it only to a different aggregate on the same File Server machine. To move a locally mounted DCE LFS fileset to a different server machine, remove its local mount point before attempting to move it. Also, because the backup version of a DCE LFS fileset is removed when the read/write version is moved, you cannot move a fileset, not even to another aggregate on the same File Server machine, if its backup version is mounted locally; you must remove the backup version's local mount point before moving the fileset.

To move the read/write version of a DCE LFS fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** files on both the source and destination machines, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entries for the source machine, the destination machine, and any machines on which replicas of the fileset reside. In addition, the source machine (specified with the **-fromserver** option) must be listed in the **admin.ft** file on the destination machine (specified with the **-toserver** option). If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Enter the **fts move** command to move a read/write fileset from one site to another:

   ```
   $ fts move -fileset {name | ID} -fromserver source_machine
       -fromaggregate source_name -toserver dest_machine
       -toaggregate dest_name
   ```

   The **-fromserver** *source_machine* option is the DCE pathname of the File Server machine on which the fileset is currently stored.

   The **-fromaggregate** *source_name* option is the aggregate on which the fileset is currently stored.

   The **-toserver** *dest_machine* option is the DCE pathname of the File Server machine on which the fileset is to be stored after moving.

   The **-toaggregate** *dest_name* option is the aggregate on which the fileset is to be stored after moving.

3. Enter the **fts lsfldb** command to confirm that the move was successful:

   $ **fts lsfldb -fileset** {*name* | *ID*}

4. Moving the read/write version of a fileset automatically deletes the backup version of the fileset if it exists at the read/write fileset's previous site. You can enter the **fts clone** command to create a new backup version at the new site:

   $ **fts clone -fileset** {*name* | *ID*}

18.9 Dumping and Restoring Filesets

The **fts dump** command converts the contents of a fileset to a byte stream format that can be stored in a file. Dumping a fileset does not affect its status in the FLDB or at the site from which it is dumped. You can dump a non-LFS fileset or any of the three types of DCE LFS filesets.

Dumping is useful when you need to save a snapshot of a fileset (for example, when a fileset is removed but may later be restored). It is also useful if the read/write version of a fileset becomes corrupted; you can dump a backup or read-only version of the fileset and restore it as the read/write version, replacing the current, corrupted version.

You can perform a full or incremental dump of a fileset. A full dump of a fileset dumps the entire fileset as it currently exists. An incremental dump of a fileset dumps only those files from the fileset that have changed since a specified date and time; only those files with modification timestamps equal to or later than the specified time are dumped.

With DCE LFS filesets, you can also perform incremental dumps of only those files that have changed since a specified fileset version. Every DCE LFS fileset has a distinct version number that increments every time an operation is performed on the fileset or a file it contains (for example, when a file, directory, or ACL is modified). Each file in the fileset also has a version number. When an operation is performed on a file in a fileset, both that file and the fileset are marked with the current version number of the fileset plus one. When you do an incremental dump based on version, files

in the fileset with version numbers equal to or greater than the version number you specify are dumped. (A DCE LFS fileset's version number is recorded in its fileset header; it has the same format as a fileset ID number. Use the **fts lsheader** or **fts lsft** command to view the current version number of a DCE LFS fileset.)

The **fts restore** command restores information from a previously dumped fileset back into the file system. Although you can dump a non-LFS fileset or any of the three types of DCE LFS filesets, you can restore a dump file only as a read/write fileset. When you restore a fileset, its creation date is set to the restoration date.

You can use the **fts dump** and **fts restore** commands to dump and restore data between different types of file systems. For example, a dump file of a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of non-LFS fileset. Similarly, a dump file of a non-LFS fileset can be restored to a DCE LFS fileset or to a different type of non-LFS fileset. In any case, the contents of the dump file are translated into the appropriate format for the file system to which the file is restored. (See your vendor's documentation to verify the level of support for dump and restore operations between different types of file systems.)

Note that incompatible information may be lost when a fileset is dumped and restored between different types of file systems. For example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs.

You can restore a dump file as a new DCE LFS fileset by specifying a name and site (File Server machine and aggregate) for the new fileset. The fileset is assigned an entry in the FLDB, and it receives the name that you specify with the command. The dump file must contain the full dump of a fileset if it is to be restored as a new fileset. After the fileset is restored, use the **fts crmount** command to create a mount point for the fileset, which makes it visible in the DCE namespace.

You can also restore a dump file as an existing read/write fileset (DCE LFS or non-LFS) by specifying the name and site of the existing fileset that is to be overwritten. The contents of the dump file overwrite the contents of the existing fileset. Include the **-overwrite** option with the **fts restore** command to specify that the existing fileset is to be overwritten; if you omit the **-overwrite** option, the command displays an error message and exits instead of overwriting the fileset. A non-LFS fileset must exist before the non-LFS partition on which it resides can be exported to the DCE namespace; therefore, when restoring a dump file as a non-LFS fileset, you

must use the **-overwrite** option to overwrite the existing non-LFS fileset, even if the fileset to be overwritten contains no data.

If you are overwriting an existing fileset with an incremental dump, the fileset to be overwritten should initially have been restored as a new read/write fileset from a full dump. Also, both the dump file that is to be restored and the full dump that initially produced the read/write fileset that is to be overwritten must be dumps of the same fileset. Note that a full dump can be restored to overwrite an existing fileset, but the restored dump file overwrites all of the data in the existing fileset; an incremental dump cannot be restored to overwrite an existing fileset that was not created from the restoration of a full dump.

Multiple incremental dumps of a fileset can be restored to overwrite the same existing fileset provided the following conditions are true:

- The fileset that is to be overwritten must not have been modified since its most recent restoration.
- The dump file that is to be restored must have been created *from* a date and time (as specified with the **-date** or **-version** option of the **fts dump** command) *no later* than the date and time at which the most recently restored dump of the fileset that is to be overwritten was dumped.
- The dump file that is to be restored must have been created *at* a date and time *later* than the date and time at which the most recently restored dump of the fileset that is to be overwritten was dumped.

The last two conditions specify that the span of time recorded in the incremental dump that is to be restored must overlap and extend the span of time recorded in the fileset that is to be overwritten. For example, suppose a full dump of a fileset was made on 1 February, an incremental dump from 31 January was made on 7 February, and a second incremental dump from 6 February was made on 14 February. The only possible way to restore all three dump files is to restore the full dump to a new read/write fileset, overwrite the new fileset with the incremental dump made on 7 February, and then overwrite the fileset with the incremental dump made on 14 February. Other sequences of restore operations involving all three dumps are very likely to result in some or all of the data in the fileset being inaccessible or inconsistent.

When restoring a dump file as a DCE LFS fileset, you can use the **-ftid** option to specify the fileset ID number that is to be associated with the fileset. If you are restoring to a new DCE LFS fileset, omit the **-ftid** option to let the FL Server allocate a new ID number; if you are overwriting an

existing DCE LFS fileset, omit the option to retain the fileset's current ID number. Specify a fileset ID number only if you are sure you can specify an unused ID.

When restoring a dump file as a non-LFS fileset, do not use the **-ftid** option. Omit the option to continue to use the fileset ID number specified for the non-LFS fileset in the entry for the partition on which the fileset resides in the **dfstab** file. Note that the restored dump file overwrites all data on the non-LFS partition.

Note: The contents of a fileset are unavailable during a dump operation. For this reason, you may want to dump only the backup version of a fileset, which does not interrupt access to the read/write and read-only versions.

18.9.1 Dumping a Fileset

To dump a fileset, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset is stored. If necessary, issue **bos lsadmin** to verify the members of the administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which you wish to store the dump file. If necessary, issue the **acl_edit -l** command to list the permissions for the directory.
3. Enter the **fts dump** command to dump the fileset:

   $ **fts dump -fileset** {*name* | *ID*} {**-time** {*date* | **0**} | **-version** *number*}
   [**-server** *machine*] [**-file** *filename*]

   The **-time** *date* or **0** option specifies a full or incremental dump. Use the **-time** option or use the **-version** option. There are three valid entries for the **-time** option:

   - The **0** entry causes a full dump of the current version.
   - The *mm/dd/yy* entry causes an incremental dump from 12:00 a.m. on the indicated date.

- The *mm/dd/yy hh:mm* entry causes an incremental dump from the specified time on the indicated date. The time must be in 24-hour format (for example, **20:30** for 8:30 p.m.). Surround the entire argument with double quotes because it contains a space (for example, **"11/22/91 20:30"**).

The **-version** *number* option specifies an incremental dump of the indicated version of the fileset. A fileset's version number is incremented with every change to the fileset or a file that it contains. Use the **-version** option or use the **-time** option. The **-version** option can be used *only* with DCE LFS filesets.

The **-server** *machine* option names the File Server machine that houses the version of the fileset to be dumped. This option is useful for dumping a particular read-only replica of a DCE LFS fileset for which multiple replicas exist.

The **-file** *filename* option specifies the complete pathname of the file in which the dump is to be stored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is sent to standard output (**stdout**).

18.9.2 Restoring a Dump File to a New Fileset

To restore a dump file to a new fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to be stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset is to be stored. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **r** (read) ACL permission for the dump file and the **w** (write), **x** (execute), and **i** (insert) ACL permissions for the directory in which the mount point for the new fileset is to be created. If necessary, issue the **acl_edit -l** command to list the permissions for the objects.

3. Select a site (an aggregate on a File Server machine) for the fileset. If necessary, enter the **fts aggrinfo** command to check the available space on the aggregate on which the fileset is to be placed:

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

4. Enter the **fts restore** command to restore the dump file to a new fileset:

   $ **fts restore -ftname** *name* **-server** *machine* **-aggregate** *name* [**-file** *filename*] [**-ftid** *ID*]

   The **-ftname** *name* option specifies the name to be assigned to the restored fileset. The name can be no longer than 102 characters. (See Chapter 17 for more information on fileset naming conventions.)

   The **-file** *filename* option specifies the complete pathname of the file that is to be restored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is taken from standard input (**stdin**).

   The **-ftid** *ID* option specifies the fileset ID number that is to be assigned to the fileset. If this option is omitted, a new ID number is allocated for the fileset. Use this option with great care; make sure the fileset ID number that you specify is not in use.

5. Issue the **fts crmount** command to create a mount point in the file system for the new fileset, making its contents visible to other users:

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

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

18.9.3 Restoring a Dump File by Overwriting an Existing Fileset

To restore a dump file by overwriting an existing fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset is to be stored, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be overwritten is stored. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **r** (read) ACL permission for the dump file. If necessary, issue the **acl_edit -l** command to list the permissions for the file.
3. Enter the **fts restore** command to restore the dump file over an existing read/write fileset, using the **-overwrite** option to overwrite the current contents of the fileset:

   $ **fts restore -ftname** *name* **-server** *machine* **-aggregate** *name*
   [**-file** *filename*] [**-ftid** *ID*] [**-overwrite**]

   The **-ftname** *name* option specifies the name of the fileset that is to be overwritten.

   The **-file** *filename* option specifies the complete pathname of the file that is to be restored. The current working directory is used if a complete pathname is not provided. If this option is omitted, the data is read from standard input (**stdin**).

   The **-ftid** *ID* option specifies the fileset ID number that is to be assigned to the fileset. If this option is omitted, the current ID number of the existing fileset is retained. Use this option with great care; make sure the fileset ID number you specify is not in use. Use this option *only* when restoring a dump file as a DCE LFS fileset; omit this option when restoring a dump file as a non-LFS fileset.

   The **-overwrite** option specifies that the restored fileset can overwrite an existing fileset. If this option is omitted, the command refuses to overwrite an existing fileset. You must use this option to overwrite a previously restored version of a fileset with a dump file that contains an incremental dump of the same fileset or to restore a dump file as a non-LFS fileset.

4. If read-only copies of the former read/write fileset exist, use the **fts update** command to replace them with replicas of the new fileset. If a backup version exists, use the **fts clone** command to replace it with a backup version of the new fileset.

18.10 Removing DCE LFS Filesets

You can use the **fts delete** command to remove read/write and backup DCE LFS filesets. You can use the **fts rmsite** command to remove replication sites and instruct the Replication Servers at the sites to remove the read-only DCE LFS filesets. You can remove a read/write version without removing its read-only versions, and vice versa. You can also remove the backup version without removing the read/write version by including the **.backup** extension on the name of the fileset that is to be removed with the **fts delete** command. However, the backup version is automatically removed when the read/write version is removed. (Note that it is possible for a backup version to remain after its read/write source is deleted if a delete operation is interrupted prior to completion.)

If no other versions of any kind exist when you remove the last version of a DCE LFS fileset, the entire FLDB entry for the fileset is removed. When you remove the last version of a DCE LFS fileset, you also need to remove its mount point with the **fts delmount** command so users do not continue to try to access the fileset's contents. It is often better to remove a fileset's mount point before deleting the fileset; removing the mount point first ensures you that no users are accessing the fileset when it is deleted.

If you remove a read/write version of a DCE LFS fileset and read-only copies still exist, the FLDB status flags for the read/write version and the backup version change to `invalid`. The fileset ID is still recorded for each type, so you can restore the read/write version later. However, when you remove a read/write version of a fileset with the **fts delete** command, you should normally also remove its read-only copies by removing its replication sites with the **fts rmsite** command.

If you remove a read-only DCE LFS fileset while other read-only sites still exist, the site information for the removed copy is deleted from the FLDB entry. If no other read-only sites exist, the status flag for the read-only version is also changed to `invalid`; the fileset ID is still recorded, so you

can re-create the read-only version later. If no other versions exist, the entire FLDB entry for the fileset is removed.

Removing the backup version of a DCE LFS fileset frees the space that it occupied on disk and changes the backup status flag in the FLDB to `invalid`. Its fileset ID is still recorded, since under normal circumstances the backup version cannot be the last existing version. When you remove the backup version, you may also want to remove its mount point from the file system.

The **fts delete** command can be used only when an FLDB entry and a fileset header exist for the fileset. Other commands can be used to remove individual FLDB entries and fileset headers. (See Section 18.10.2 for more information about these commands.)

Note that when you delete a read/write or backup version of a DCE LFS fileset, that version of the fileset no longer exists on disk. Before removing the read/write or backup version of a fileset, use the DFS Backup System to preserve a permanent copy of it on tape.

Note: You cannot remove a DCE LFS fileset that is also mounted locally (as a file system on its File Server machine). You must remove the fileset's local mount point before attempting to delete the fileset. Also, because the backup version of a fileset is removed when the read/write version is removed, you cannot remove the read/write version of a fileset if its backup version is mounted locally; you must remove the backup version's local mount point before deleting the read/write version.

18.10.1 Removing a DCE LFS Fileset and Its Mount Point

To remove a DCE LFS fileset and its mount point, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.ft** file on the machine on which the fileset resides, and you must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Verify that you have the **w** (write), **x** (execute), and **d** (delete) ACL permissions for the directory in which the fileset is mounted. If necessary, issue the **acl_edit -l** command to list the permissions for the directory.

3. Issue the **fts rmsite** command to remove the fileset's replication sites and to instruct the Replication Servers at the sites to remove the read-only versions of the fileset:

   $ **fts rmsite -fileset** {*name* | *ID*} **-server** *machine* **-aggregate** *name*

   Repeat the **fts rmsite** command to remove each of the fileset's replication sites. If Release Replication was used for the fileset, the **fts rmsite** command must also be used to remove the replication site and read-only replica at the read/write fileset's site.

4. Issue the **fts delete** command to remove the read/write and backup versions of the fileset:

   $ **fts delete -fileset** {*name* | *ID*} **-server** *machine* **-aggregate** *name*

5. After removing the last copy of the fileset, enter the **fts delmount** command to remove the mount point. Disregard this step if any copies of the read-only version remain in the file system and you want them to be accessible.

   $ **fts delmount -dir** *directory_name...*

   The **-dir** *directory_name* option is the name of each mount point that you want to remove.

   If you previously mounted the backup version as a subdirectory of the read/write fileset's root directory, removing the read/write version's mount point makes the backup version's mount point inaccessible. If you mounted the backup version at a separate directory, you must explicitly remove the backup version's mount point, again using the **fts delmount** command.

18.10.2 Other Commands for Removing Filesets

Under normal circumstances, always use the **fts delete** or **fts rmsite** command to remove a fileset; these commands automatically record the deletion in the FLDB. Under special circumstances, however, you may need to use the following commands. Keep in mind that if the FLDB and the filesets are consistent with each other, these commands make them inconsistent. Never use these commands unless absolutely necessary.

- Use the **fts delfldbentry** command to remove an FLDB entry that mentions a particular fileset. If versions of the fileset still exist at sites, they are not affected. This is useful if you are certain that a fileset removal was not recorded in the FLDB, and you do not want to use the **fts syncfldb** and **fts syncserv** commands to synchronize the entire FLDB. Use the **fts lsfldb** or **fts lsft** command to determine if a fileset removal was recorded in the FLDB.
- Use the **fts zap** command when it is urgent that a fileset be removed from its site, but the FLDB is inaccessible (for example, if the FL Server is unavailable). You can then remove the fileset's entry from the FLDB by entering the **fts delfldbentry** command or by entering the **fts syncfldb** and **fts syncserv** commands to synchronize the FLDB. The **fts zap** command, like the **fts delete** command, cannot be used to remove a DCE LFS fileset that is also mounted locally; you must remove the fileset's local mount point before attempting to delete the fileset.

The following subsections provide brief descriptions of the syntax and use of these commands.

18.10.2.1 Removing a Fileset's FLDB Entry Without Removing the Fileset

To remove a fileset's FLDB entry without removing the fileset, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine that houses a version of any fileset whose FLDB entry is to be removed. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Enter the **fts delfldbentry** command to remove the fileset entry from the FLDB. Because this command lets you remove multiple FLDB entries simultaneously, be careful to remove only those FLDB entries you no longer need.

   $ **fts delfldbentry** {**-fileset** {*name* | *ID*} | **-prefix** *string*} [**-server** *machine*] [**-aggregate** *name*]

   The **-prefix** *string* option specifies a character string of any length. Every FLDB entry that lists a fileset whose name begins with *string* is removed. If the **-server** and **-aggregate** options are specified, only entries for filesets on the specified server machine and the specified aggregate are removed. Use the **-prefix** option or use the **-fileset** option to specify the name or ID number of the fileset whose FLDB entry is to be removed.

18.10.2.2 Removing a DCE LFS Fileset Without Updating Its FLDB Entry

To remove a DCE LFS fileset without updating its FLDB entry, do the following:

1. Verify that you have the necessary privilege. You must be included in the **admin.ft** file on the machine on which the fileset to be removed resides. If necessary, issue the **bos lsadmin** command to verify the members of the administrative list.
2. Enter the **fts zap** command to remove the fileset without recording the removal in the FLDB:

   $ **fts zap -ftid** *ID* **-server** *machine* **-aggregate** *name*

   The **-ftid** *ID* option specifies the fileset ID number of the fileset that is to be removed.

18.10.3 Removing Non-LFS Filesets

When you remove a non-LFS fileset, it becomes inaccessible in the DCE namespace. However, it is still available on the local disk of the machine on which it resides. (You can use the appropriate command in your local operating system to remove the partition that houses the non-LFS fileset from the disk.)

To remove a non-LFS fileset from the DCE namespace, use the **fts delfldbentry** command to remove the entry for the fileset from the FLDB. This prevents the FL Server from reporting the location of the fileset to a Cache Manager that requests data from the fileset. The **fts delfldbentry** command lets you remove multiple FLDB entries simultaneously; be careful to remove only those FLDB entries you no longer need.

Once you remove the fileset's FLDB entry, use the **fts delmount** command to remove the mount point for the fileset. Then issue the **dfsexport** command with the **-detach** option to detach the non-LFS partition on which the fileset resides from the namespace; when you detach a partition, it is no longer exported. These steps make the fileset unavailable in the DCE namespace. After you issue the **dfsexport** command, remove the partition's entry from the *dcelocal*/**var/dfs/dfstab** file to prevent it from being exported the next time the machine is rebooted; note that this occurs only if the **dfsexport** command is included in the machine's initialization file (**/etc/rc** or its equivalent).

Any of these steps performed alone makes the fileset inaccessible. However, you should always perform all of the steps whenever you remove a non-LFS fileset to prevent future problems, such as a mount point that references a fileset that is no longer exported.

To remove a non-LFS fileset and its mount point, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for the machine on which the fileset resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Verify that you have the **w** (write), **x** (execute), and **d** (delete) ACL permissions for the directory in which the fileset is mounted. If necessary, issue the **acl_edit -l** command to list the permissions for the directory.

3. Issue the **fts delfldbentry** command to remove the fileset entry from the FLDB:

   $ **fts delfldbentry** {**-fileset** {*name* | *ID*} | **-prefix** *string*} [**-server** *machine*] [**-aggregate** *name*]

   The **-prefix** *string* option specifies a character string of any length. Every FLDB entry that lists a fileset whose name begins with *string* is removed. If the **-server** and **-aggregate** options are specified, only entries for filesets on the specified server machine and the specified aggregate are removed. Use the **-prefix** option or use the **-fileset** option to specify the name or ID number of the fileset whose FLDB entry is to be removed.

4. Enter the **fts delmount** command to remove the fileset's mount point:

   $ **fts delmount -dir** *directory_name...*

   The **-dir** *directory_name* option is the name of each mount point that you want to remove.

5. Log in as **root** on the machine on which the fileset resides.

6. Enter the **dfsexport** command with the **-detach** option to detach the partition from the DCE namespace:

   # **dfsexport -aggregate** *name* **-detach**

   The **-aggregate** *name* option specifies the device name or exported aggregate name of the partition to be detached.

   The **-detach** option indicates that the specified partition is to be detached.

7. Use a text editor to remove the partition's entry from the **dfstab** file. An entry for a partition in the **dfstab** file has the following format:

   ```
   /dev/lv02  /usr  ufs  1  0,,18756
   ```

18.11 Locking and Unlocking FLDB Entries

The FL Server locks the FLDB entry for a DCE LFS or non-LFS fileset before the Fileset Server executes any operations on it. A fileset with a locked FLDB entry is not affected by any other fileset manipulation operations such as moving or backing up a fileset. This immunity from other operations prevents inconsistencies or corruptions that can result from multiple simultaneous operations on a fileset. You can use the **fts lock** command to lock an FLDB entry and prevent an **fts** command from accessing it. You may want to lock an entry when you suspect it may be in error to prevent anyone from writing to it while you check the problem.

If an **fts** command operation fails prematurely, the FLDB entries can remain locked, preventing you from executing commands that can correct the problems. You can use the **fts lsft** and **fts lsfldb** commands to examine locked FLDB entries; you can use the **fts unlock** command to unlock a specific FLDB entry.

The **fts unlockfldb** command lets you unlock a set of entries based on the options you provide. When you

- Provide no options, all locked entries are unlocked
- Specify a File Server machine with the **-server** option, all locked entries with that machine in a site definition are unlocked
- Specify an aggregate with the **-aggregate** option and a File Server machine with the **-server** option, all locked entries with that aggregate and that machine in a site definition are unlocked

18.11.1 Determining Whether an FLDB Entry Is Locked

To determine whether an FLDB entry is locked, issue the **fts lsfldb** command:

$ fts lsfldb -fileset {*name* | *ID*}

If the entry is locked, the word Locked appears on a line of the output of the command. The **fts lsft** command also displays the same line in its output if an entry it is examining is locked.

18.11.2 Locking an FLDB Entry

To lock an FLDB entry, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be locked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts lock** command to lock the entry:

   $ **fts lock -fileset** {*name* | *ID*}

18.11.3 Unlocking a Single FLDB Entry

To unlock a single FLDB entry, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of the fileset to be unlocked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts unlock** command to unlock the entry:

   $ **fts unlock -fileset** {*name* | *ID*}

18.11.4 Unlocking Multiple FLDB Entries

To unlock multiple FLDB entries, do the following:

1. Verify that you have the necessary privileges. You must be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine on which a version of any fileset to be unlocked resides. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.

2. Issue the **fts unlockfldb** command to unlock all entries or only those entries on a specified server, on a specified aggregate, or on both:

   $ **fts unlockfldb** [**-server** *machine*] [**-aggregate** *name*]

18.12 Synchronizing the FLDB and Fileset Headers

In DFS, transparent file access is possible because the FLDB constantly tracks fileset locations. When the Cache Manager needs a file, it contacts the FL Server, which consults the FLDB to find the current location of the file. Therefore, the FLDB must accurately reflect the state of filesets on all File Server machines. To keep the FLDB accurate, all **fts** commands that affect fileset status automatically record the change in the appropriate FLDB entry.

Whenever you issue a command that changes the status of a fileset, the **fts** program directs the

- FL Server to lock the FLDB entry; the lock advises other operations not to attempt to manipulate any of the fileset's versions (read/write, read-only, or backup). This prevents simultaneous operations.
- FL Server to set an intention flag in the FLDB entry to indicate the type of operation to be performed.
- Fileset Server to perform the operation on the fileset. It may set an `Off-line` flag in the header, making the fileset inaccessible to other operations. When the operation is completed, the fileset is again marked `On-line`.
- FL Server to record the changes from the Fileset Server's operation in the FLDB. When the operation completes, the lock is released and the intention flag is removed. The fileset is again available for further operations.

Errors can occur if you are forced to stop an operation with an abort signal or if a File Server machine or server process goes down after you issue an **fts** command but before the requested operation is complete. It is likely in these situations that the FLDB is not synchronized with the headers of filesets on File Server machines.

The following symptoms indicate that the FLDB and fileset headers are not synchronized:

- Error messages indicate that the operation terminated abnormally.
- A subsequent **fts** operation fails because the initial failure left an FLDB entry locked.
- A subsequent **fts** operation fails because the initial failure left a fileset marked `Off-line`.

The **fts syncfldb** and **fts syncserv** commands are used to synchronize the FLDB and fileset headers. The **fts syncfldb** command examines the fileset header of each online fileset on a File Server machine (and optionally aggregate) that you specify. It checks the FLDB entry that is associated with the fileset to verify that the FLDB correctly records the fileset's status at the site. If the FLDB is incorrect, **fts syncfldb** alters it; if no entry exists for an online fileset, it creates one. The command also performs two additional functions:

- If it finds a backup fileset whose read/write source no longer exists at the same site, it removes the backup from the site.
- If it finds a fileset ID number that is larger than the value of the counter that is used by the FL Server when allocating fileset ID numbers, it records this ID number as the new value of the counter. The next fileset to be created receives a fileset ID number that is one greater than this number.

The **fts syncserv** command examines every FLDB entry that mentions a site on a File Server machine (and optionally aggregate) that you specify. It verifies that the FLDB information is consistent with the state of the fileset at the site. The command also solves the following types of discrepancies:

- If a fileset that appears to be normal in the FLDB is marked `Off-line` on the File Server machine, it brings the fileset online.
- If a fileset version that is recorded in the FLDB does not exist at the indicated site, it changes the FLDB entry. If the site is for the read/write, backup, or the last read-only version, it sets the status flag for the version to `invalid`.
- If a fileset's status in its FLDB entry indicates that it should not exist at its site, it removes the fileset. This can occur if the fileset remains from an aborted or failed fileset operation.

To ensure that the FLDB and all fileset headers in your cell are synchronized, run the **fts syncfldb** command once for each File Server machine in your cell. Then run the **fts syncserv** command once for each File Server machine in your cell.

Note that while the **fts syncfldb** and **fts syncserv** commands are useful in error recovery, they do not, in general, recover all of the information that is stored with a fileset's entry in the FLDB. More specifically, if the FLDB entry for a DCE LFS fileset is removed somehow and then re-created with the **fts syncfldb** command, replication information associated with the fileset is not restored. The **fts syncfldb** and **fts syncserv** commands cannot reproduce replication information once the entry for a DCE LFS fileset is removed from the FLDB. You must use the **fts setrepinfo** and **fts addsite** commands to reconstruct the replication information.

18.12.1 Synchronizing Non-LFS Filesets

The **fts syncfldb** and **fts syncserv** commands can be used to ensure the consistency of non-LFS filesets. However, because non-LFS filesets do not have fileset headers, the effectiveness of the commands is limited. You may need to take a more active role in returning consistency to non-LFS filesets and their FLDB entries.

For example, because non-LFS filesets do not have fileset headers, the **fts syncfldb** command cannot determine the name of a non-LFS fileset that has no FLDB entry. If the command determines that it needs to create an FLDB entry for a non-LFS fileset, it generates a unique name of the form **SYNCFLDB-ADDED-***number*. You then need to enter the proper commands to rename the fileset to its original name.

Similarly, because the **fts syncserv** command cannot destroy a disk partition, it cannot delete a non-LFS fileset, even if it determines that the fileset needs to be deleted. Instead, the **fts** program displays an error message reporting the non-LFS fileset that needs to be deleted to restore file system consistency. You must then enter the proper commands to remove the fileset from the DCE namespace.

18.12.2 Synchronizing Fileset Information

To synchronize the FLDB and fileset headers, do the following:

1. Verify that you have the necessary privileges. You must included in the **admin.ft** file on each machine that houses a version of any fileset stored at a site specified with the **fts syncfldb** or **fts syncserv** command. You must also be included in the **admin.fl** file on each Fileset Database machine or own the server entry for each machine that houses a version of any fileset stored at a site specified with either command. If necessary, issue the **bos lsadmin** command to verify the members of an administrative list.
2. Issue the **fts syncfldb** command to make FLDB entries consistent with filesets that are stored at the specified site. *Repeat this command for every File Server machine in your cell.*

   $ **fts syncfldb -server** *machine* [**-aggregate** *name*]

   The **-aggregate** *name* option names the aggregate on the server machine specified with the **-server** option for which fileset headers and FLDB entries are to be compared. Do not use this option under normal circumstances; omitting it allows synchronization of all of the filesets on the machine specified with the **-server** option. Use it only when just a single aggregate needs to be synchronized.
3. Issue the **fts syncserv** command to make filesets that are stored at the specified site consistent with FLDB entries. *Repeat this command for every File Server machine in your cell.*

   $ **fts syncserv -server** *machine* [**-aggregate** *name*]

   The **-aggregate** *name* option names the aggregate on the server machine specified with the **-server** option for which fileset headers and FLDB entries are to be compared. Do not use this option under normal circumstances; omitting it allows synchronization of all of the filesets on the machine specified with the **-server** option. Use it only when just a single aggregate needs to be synchronized.

18.13 Verifying and Maintaining File System Consistency

Many operating systems use the **fsck** program to ensure file system consistency after a system failure. The **fsck** program checks the consistency of a file system and reports its findings. Optionally, it repairs problems that it finds in the file system. The **fsck** program (or its equivalent) is still used to return consistency to many types of non-LFS partitions.

DFS employs a log mechanism and an additional system application, the DFS Salvager, to ensure the consistency of DCE LFS aggregates. A log is kept of all changes made to metadata on a DCE LFS aggregate as a result of operations such as file creation and deletion. The metadata records the structure and organization of the file system. Each DCE LFS aggregate has its own log, which physically resides on the aggregate, where it is completely transparent to users.

The DFS Salvager returns consistency to a file system when the system is restarted by replaying the log. Under normal circumstances, replaying the log returns the file system to a consistent state. However, if the Salvager detects problems in the basic structure of the aggregate, if the log mechanism is damaged, or if the physical storage medium of the aggregate is suspect, replaying the log cannot restore consistency. In these cases, a system administrator must invoke the Salvager a second time to examine and repair the structure of the aggregate.

18.13.1 Overview of the DFS Salvager

The DFS Salvager is used to replay the log on an aggregate and, if necessary, to find and repair file system inconsistencies that cannot be repaired by replaying the log. Along with the normal consistency guarantees provided by the log mechanism, the Salvager performs the same type of functions as the **fsck** program in other operating systems: it reads the metadata that describes the contents of a file system, analyzes the internal organization and structure of the file system, and detects and repairs inconsistencies.

The Salvager is invoked with the *dcelocal*/**bin/salvage** command. The command can be used to direct the Salvager to do the following:

- Recover an aggregate following a system restart by replaying the log on the aggregate. The Salvager's replaying of the log is referred to as running recovery on the aggregate (or simply recovering the aggregate). This is the normal production use of the Salvager. Unless the contents of the log or the physical structure of the aggregate is damaged, replaying the log is an effective guarantee of a file system's integrity.
- Verify the structure of an aggregate to determine if it contains any inconsistencies. Recovering an aggregate and then verifying its structure represents a cautious application of the Salvager. Note that the Salvager can also be used to verify an aggregate before recovery is run, but it typically finds problems with an unrecovered aggregate that it would not find were recovery run first.
- Salvage an aggregate by attempting to repair any inconsistencies it finds in the structure of the aggregate. Because recovery eliminates inconsistencies in an undamaged file system, an aggregate is typically recovered before it is salvaged. It is usually a good idea first to recover and then to salvage an aggregate if a machine panics or experiences a hardware failure.

As noted, running recovery to return consistency to a file system at restart time is the normal application of the Salvager. When it is installed, DFS automatically updates the local **/etc/rc.dfs** configuration file to include the commands necessary to recover each DCE LFS aggregate listed in the **dfstab** file when the system is restarted. The system administrator can use the Salvager to verify or salvage an aggregate in addition to or instead of running recovery, as the situation warrants.

It is important to distinguish between file system consistency and user data consistency. The Salvager reads file system metadata; it does not try to verify the contents of the files in that file system. The Salvager can verify that each block in a file is correctly attached to that file, but it cannot verify the actual contents of the blocks. In cases where the metadata associated with a file is damaged, the owner of the file needs to verify that the file's contents are intact.

For example, if a disk controller accidentally writes on the disk surface, the Salvager tries to repair any inconsistencies in the structure of the file system. However, it has no mechanism to guarantee the contents of any file. In this case, the Salvager identifies any files whose metadata was damaged.

After the file system is salvaged, users can verify the contents of these files; files whose contents were damaged can be restored from backups made before the file system problems occurred.

Not all aggregates can be salvaged. In the case of extensive damage to the structure of an aggregate or damage to the physical disk that houses an aggregate, the Salvager cannot repair inconsistencies.

18.13.2 Differences Between the DFS Salvager and fsck

While the DFS Salvager performs a role similar to that of the **fsck** program, several major differences distinguish the two programs:

- Recovery is usually sufficient to restore file system consistency at boot time. For this reason, the Salvager is typically used only to recover, not to verify or salvage, an aggregate when the system reboots. An administrator needs to use the Salvager to salvage an aggregate only if file system damage is suspected; for example, if the log cannot be replayed successfully, if a fileset cannot be mounted, or if a controller or disk failure affects the file system.
- The Salvager does not normally prompt the issuer for direction. It asks for confirmation to proceed only if it suspects that the aggregate on which it is run is not a DCE LFS aggregate or if it finds that the size of the aggregate that is recorded on disk exceeds the capacity of the partition on which the aggregate resides. It never asks the issuer for direction on how to repair the file system, in which respect it is similar to the **fsck -p** command. Because of this, it can be run in the background, and several Salvager processes can be run simultaneously.

  Note that the **-force** option of the **salvage** command can be used to direct the Salvager to proceed with all operations without requesting confirmation. However, if the Salvager is run on an invalid aggregate, using the **-force** option can produce unexpected changes.
- The Salvager displays information about files to be restored based on problems it discovers when it verifies or salvages an aggregate. A complete list of files (with pathnames, if possible) is printed when the operation completes. This output helps the system administrator complete the recovery of the files in the repaired file system.

18.13.3 Using the DFS Salvager

The **salvage** command is used to direct the Salvager to recover, verify, or salvage the structure of an aggregate. Combine the command's **-recoveronly**, **-verifyonly**, and **-salvageonly** options as follows to specify the operations the Salvager is to perform on the specified aggregate:

Specify the **-recoveronly** option

To run recovery on the aggregate without attempting to determine or repair any inconsistencies found on it. Use this option to quickly return consistency to an aggregate that does not need to be salvaged. This represents the normal production use of the Salvager.

Specify the **-verifyonly** option

To determine whether the structure of the aggregate contains any inconsistencies without running recovery or attempting to repair any inconsistencies found on the aggregate. Use this option to assess the extent of the damage to an aggregate. The Salvager makes no modifications to an aggregate during verification. Note that it is normal for the Salvager to find errors when it verifies an aggregate that has not been recovered.

Specify the **-recoveronly** and **-verifyonly** options

To run recovery on the aggregate and then analyze its structure without attempting to repair any inconsistencies found on it. Use these options if you believe that replaying the log can return consistency to the aggregate, but you want to verify the consistency of the aggregate after recovery is run. This approach is more cautious than recovering the aggregate without verification.

Specify the **-salvageonly** option

To attempt to repair any inconsistencies found in the structure of the aggregate without first running recovery on it. Use this option if you believe that the log is damaged or that replaying the log will not return consistency to the aggregate and may in fact further damage it. Under normal circumstances, do not salvage an aggregate without first recovering it.

Omit the **-recoveronly**, **-verifyonly**, and **-salvageonly** options

To run recovery on the aggregate and then attempt to repair any inconsistencies found in the structure of the aggregate. Omit these three options if you believe the log should be replayed before attempts are made to repair any inconsistencies found on the aggregate.

The Salvager cannot be used to recover or salvage an aggregate that is currently exported to the DCE namespace. If asked to perform either of these operations on an exported aggregate, the Salvager exits without performing the operation. If necessary, use the **dfsexport** command to detach an aggregate from the global namespace before recovering or salvaging it. Note that the Salvager also exits if it is run on an aggregate that houses a locally mounted fileset.

The Salvager prompts for direction only if

- The aggregate on which it is run contains a non-LFS superblock whose creation time is more recent than that of its DCE LFS superblock.
- The size of the aggregate that is recorded in its DCE LFS superblock exceeds the capacity of the partition on which the aggregate resides.

At the prompt, the issuer can choose to cancel or continue the operation. If the operation is continued in either of these situations and the aggregate proves to be invalid, the operation can produce unpredictable results. The best response in either case is to cancel the operation and attempt to determine the cause of the problem. Note that the command's **-force** option can be used to direct the Salvager to proceed rather than prompt for confirmation in these cases.

Internal structures maintained by the Salvager require a minimum of 1 megabyte of swap space. However, the total amount of swap space required by the Salvager depends largely on the size of the aggregate being salvaged and the extent of the damage to the aggregate.

18.13.4 Recovering, Verifying, or Salvaging a File System

To recover, verify, or salvage a file system, do the following:

1. If the specified aggregate is to be recovered or salvaged, log in as **root** on the local machine or verify that you have both the **r** (read) and **w** (write) permissions for the aggregate. If the specified aggregate is to be verified, log in as **root** on the local machine or verify that you have the **r** (read) permission for the aggregate.
2. Ensure that the DCE LFS aggregate to be specified with the command is not exported and contains no locally mounted filesets.
3. Enter the **salvage** command to run recovery on the aggregate, verify the consistency of the aggregate, or attempt to repair the consistency of the aggregate:

   $ **salvage -aggregate** *name* [**-recoveronly**] [{**-verifyonly** | **-salvageonly**}] [**-force**] [**-verbose**]

   The **-aggregate** *name* option is the device name or aggregate name of the DCE LFS aggregate that is to be recovered, verified, or salvaged. Note that the aggregate ID of the aggregate is not an acceptable value.

   The **-recoveronly** option directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate.

   The **-verifyonly** option directs the Salvager to verify the specified aggregate. The Salvager examines the structure of the aggregate to determine if it contains any inconsistencies, reporting any that it finds.

   The **-salvageonly** option directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate.

   The **-force** option executes the Salvager in noninteractive mode. By default, the Salvager prompts for confirmation before proceeding in certain situations. Use this option to direct the Salvager to proceed with all operations without asking whether it should continue.

   The **-verbose** option directs the Salvager to produce detailed information about the aggregate as it executes. The information is useful primarily for debugging purposes. Use this option alone or with any other combination of options.

18.13.5 Interpreting Salvager Output

The Salvager displays output on the screen. When the **salvage** command is first executed, the Salvager displays the device name of the aggregate on which it is to run and the operation it is to perform. For example, it displays the following message if it is instructed to recover an aggregate:

```
Will run recovery on device
```

If the Salvager is used to recover an aggregate and the log on the aggregate does not need to be replayed, the Salvager displays no further output. If the log does need to be replayed and the Salvager can successfully recover the aggregate, the Salvager displays the following messages:

```
Recovery statistics
    statistics
Ran recovery on device
```

In the output, *statistics* consists of a few lines of information about the log and its replaying, and *device* is again the device name of the recovered aggregate. If recovery fails for any reason, the Salvager returns an appropriate exit code. (All Salvager exit codes are described at the end of this section.)

The Salvager can display much more output if it is used to verify or salvage an aggregate on which it finds metadata errors. As it verifies or salvages a damaged aggregate, it displays a message similar to the following for each fileset in which it encounters metadata problems:

```
In volume fileset (avl #integer)
    in anode (#integer)
        description
```

It displays the first line once for each fileset, repeating the second and third lines once for each problem anode in the fileset. An anode is an area on disk that provides information used to locate data such as files, directories, ACLs, and other types of file system objects. Each fileset contains an arbitrary number of anodes, all of which must reside on the same aggregate.

In the output, *fileset* is the name and ID number of each affected fileset; **avl** *#integer* indicates the anode for the fileset; **in anode** (*#integer*) indicates the anode for a file or other object in the fileset; and *description* provides a

brief description of the problem the Salvager found with the anode (and any actions it took as a result, if it is salvaging the aggregate).

When it has finished executing, the Salvager lists files whose metadata it found to be damaged, many of which it likely repaired if it salvaged the aggregate. For each file, it displays a line of the form:

condition fileset`:`*pathname* `volume index:` *integer* `anode index:` *integer*

In the output, *condition* is a string that describes the state of the file or its metadata; *fileset* is the name of the fileset in which the affected file resides; and *pathname* is the pathname of the file, relative to the root directory of the fileset. Note that the Salvager may not be able to determine the fileset name or reconstruct the pathname for every file.

The **volume index:** *integer* and **anode index:** *integer* provide pointers to the anodes that are associated with filesets and files whose metadata was damaged (and possibly repaired). The **volume index** indicates the anode for the fileset; the **anode index** indicates the anode for the file. Anode-related information is not useful for verifying or restoring data on an aggregate, but it does serve to identify earlier messages displayed by the Salvager that are related to this file.

The following conditions accompany the files most in need of attention:

oughtRestore Files in which one or more block references in the associated anode were removed or changed. Because it is unlikely that such files contain all of their original data, these files should be restored from existing backups. This condition applies only to files on salvaged aggregates.

mayRestore Files to which modifications were made (for example, files whose ACLs or property lists were changed). The owners of these files should verify their contents, or a system administrator should simply restore them from backups if a directory listing indicates that they have not been modified since the last backup was made. This condition also applies only to files on salvaged aggregates.

zeroLinkCnt Files whose link counts should be 0 (zero). These files were deleted but not closed when the system crashed or were orphaned by the Salvager as it made repairs to the file system. The system will delete them when the aggregate is exported.

badLinkCnts Files whose link counts were inconsistent with the number of references found to them. These files should be examined, if possible, or simply restored.

The Salvager can list a file more than once if it finds that multiple conditions apply to the file. It can also display one or more additional conditions, but files with which the additional conditions are associated are usually already covered by one or more of the conditions just described.

The Salvager also returns one of various exit codes summarizing its actions and findings. It returns the exit codes in the form of bits, which it uses to indicate the state of the aggregate. It can set multiple bits, but in general, the higher the bit, the greater the severity of the aggregate's problems; the higher bit always takes precedence when interpreting the output. The Salvager can return the following exit codes:

All bits off

The Salvager found no problems. It displays a message that includes `Done` and `Checks out`. The command need not be run again.

First bit (**0x1**) set

The Salvager found one or more problems. It displays a message that includes `Done` and `Some inconsistencies found`. Run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Second bit (**0x2**) set

The Salvager found one or more problems and fixed them. It displays a message that includes `Done` and `Some inconsistencies repaired`. The command need not be run again. (Note that if the second bit is set, the first bit is typically set as well; because the higher bit takes precedence, you do not need to run the command again.)

Third bit (**0x4**) set

The Salvager found one or more problems and fixed some of them. It displays a message that includes `Incomplete` and `Some repairs made`. Some of the problems were more severe and require a subsequent salvage to be repaired; run the command on the aggregate without the **-verifyonly** option to attempt to correct the problems.

Fourth bit (**0x8**) set

The Salvager found the aggregate to be irreparably damaged. It displays a message that begins `Problem`. Use the **newaggr** command to reinitialize the aggregate, and reconstruct the data from existing backups if possible.

Fifth bit (**0x10**) set

The Salvager found some serious problem that prevents it from running; for example, the attempted recovery of the aggregate failed because of damage to the log, or the attempted salvage of the aggregate failed because the aggregate is not a DCE LFS aggregate or it is currently exported. The Salvager displays a message that begins `Problem`. Attempt to determine the cause of the problem.

Including the **-verbose** option with the **salvage** command causes the Salvager to produce more detailed information about the aggregate. Much of the additional information is useful primarily for debugging purposes.

Chapter 19

Configuring the Cache Manager

This chapter describes the configuration of any machine that is to serve as a DFS client machine. All DFS client machines must do the following:

- Run the set of modifications known as the Cache Manager in the kernel. The Cache Manager enables the client to access DFS. You can control several aspects of Cache Manager and client performance by configuring the Cache Manager as described in this chapter.
- Run the **dfsd** process, which initializes the Cache Manager by transferring DFS configuration information into kernel memory.
- Run the **dfsbind** process, which resolves CDS pathnames for the Cache Manager and accesses user authentication information necessary for effective communications with server machines.
- Provide local disk space or memory sufficient to house configuration information and a cache.

19.1 An Overview of the Cache Manager

The Cache Manager fetches and caches files from File Server machines on behalf of application programs that are running on client machines. To locate a file to be retrieved, it first contacts the Fileset Location Server (FL Server) to learn the location of the fileset that houses the file; to retrieve the file, it then contacts the File Server machine that houses the file. The File Exporter on the File Server machine delivers the file, which the Cache Manager stores in the client machine's cache, which is an area of local disk or memory designated for temporary storage. Once the data is cached locally, the Cache Manager can access it as quickly as it can a local file.

The Cache Manager verifies that its cached files match the central copies at File Server machines by keeping the tokens that the File Exporters send with the files. A token acts as the File Exporter's promise to contact the Cache Manager if the centrally stored copy of a file changes while the Cache Manager has a cached copy. If the central copy changes, the File Exporter revokes the token; the Cache Manager sees that the token is revoked and retrieves the new version of the file when an application program next requests data from it.

19.1.1 Cache Manager Processes

The **dfsd** process controls the Cache Manager, which runs in the kernel. You can add options to the *dcelocal*/**bin/dfsd** command and modify the *dcelocal*/**etc/CacheInfo** file on the client machine to customize DFS cache parameters.

You can control several Cache Manager features with **dfsd** options. The **dfsd** process must be invoked at or after system reboot on all DFS client machines; it is recommended that the **dfsd** command be added to the proper initialization file (**/etc/rc** or its equivalent). The **dfsbind** process must be run before the **dfsd** process; it is recommended that the *dcelocal*/**bin/dfsbind** command also be added to the proper initialization file.

19.1.2 Cache Manager Files

The **CacheInfo** file, which is manually created during DFS client installation, is composed of three fields separated by colons: the first field specifies where the DCE global namespace is mounted; the second field names the cache directory where the Cache Manager creates its cache files; and the third field specifies the cache size in 1024-byte (1-kilobyte) blocks.

The Cache Manager creates and maintains the following files, which are not intended for direct use. You can cause the kernel to panic if you accidentally modify any of these files; if this happens, rebooting the machine should restore normal performance. Note that these files exist only on client machines that use disk caching; a machine that uses a memory cache maintains in memory the cache information the files contain.

Caution: *Never* directly modify or delete these files; this can cause the kernel to panic. *Always* use the commands provided with DFS to alter these files.

- Multiple *dcelocal*/**var/adm/dfs/cache/V***n* files, where *n* is a unique number for each file. By default, each **V***n* file holds up to 64 kilobytes of a cached file; files larger than 64 kilobytes are divided into multiple files. The number of **V***n* files, or **V** files, depends on the cache size.
- The *dcelocal*/**var/adm/dfs/cache/CacheItems** file. The Cache Manager uses this binary file to record information about each **V** file, including its file ID number and data version number.
- The *dcelocal*/**var/adm/dfs/cache/FilesetItems** file. This binary file stores the fileset-to-mount point mapping for each fileset accessed. This mapping enables the Cache Manager to respond correctly to commands such as **pwd**.

19.2 Cache Manager Features You Can Customize

You can alter the following aspects of the Cache Manager configuration to achieve different levels of performance on different client machines:

- Disk or Memory Cache: You can direct the Cache Manager to use machine memory instead of disk space for caching.
- Cache Location: The standard cache location (*dcelocal*/**var/adm/dfs/cache**) can be changed to take advantage of greater space availability on other partitions.
- Cache Size: The cache size influences how often the Cache Manager contacts File Server machines across the network. Increasing the cache size results in better performance because fewer cross-network calls are necessary.
- Chunk Size and Number: You can use several options with **dfsd** to alter the default size and number of chunks that compose a cache. With a disk cache, each chunk is called a **V** file; with a memory cache, each chunk is represented by a block of memory. The size and number of chunks can be modified to take advantage of fast networks or to compensate for slow networks.
- The **setuid** Status: By default, the Cache Manager does not allow **setuid** programs from filesets to execute with **setuid** permission. You can enable **setuid** programs from specific filesets to execute with **setuid** permission; **setgid** programs on a fileset are enabled and disabled along with **setuid** programs.
- Device File Status: By default, the Cache Manager does not honor device files stored in filesets. You can instruct the Cache Manager to recognize device files from specific filesets.
- Cached File Versions: The DFS token mechanism guarantees that the Cache Manager uses the most current versions of files and directories. You can also force the Cache Manager to discard the versions you are using and fetch new versions from the File Server machine.

- Unstored Data: If the Cache Manager cannot contact a File Server machine to write data to it, it keeps the unstored data in the cache. It then continues to attempt to contact the File Server machine until it can store the data. You can list all of the data the Cache Manager cannot store, and you can force the Cache Manager to discard the data rather than to continue to try to contact unavailable File Server machines.

Note: You must issue the commands described in this chapter at a console or terminal of the machine being configured; you cannot specify a different machine name to be used with these commands. Some of the commands require that you log in as **root**, while others require no privileges; the necessary privileges are indicated with each command. All of the files mentioned in this chapter are local files.

19.3 Choosing Cache Type, Location, and Size

The default DFS configuration is disk caching. However, the Cache Manager can use a machine memory cache rather than a disk cache. To direct the Cache Manager to use memory caching, use the **-memcache** option with the **dfsd** command. When the **-memcache** option is used, the Cache Manager does no disk caching, even if the machine has a disk available.

The **CacheInfo** file defines the directory to use for a disk cache and the size of a disk or memory cache. The Cache Manager checks this file at initialization to determine this information. (The installation instructions provide details for creating the **CacheInfo** file.) The **CacheInfo** file contains the following three fields separated by colons:

- A directory on the local disk where the Cache Manager mounts the DCE global namespace. The default entry is the global namespace designation (/...). If /... is not specified, symbolic links to the global namespace fail.
- A local directory that serves as the DFS cache for a disk cache. The Cache Manager creates its cache files in this directory. The default entry is *dcelocal*/**var/adm/dfs/cache**. Although the indicated directory is not used with a memory cache, an entry *must* appear in this field even if memory caching is employed on the machine.

- A definition of the cache size in kilobyte blocks.

Following is an example of a **CacheInfo** file. The file lists the DCE namespace mounted at the global namespace designation (/...), the *dcelocal*/**var/adm/dfs/cache** directory used for the cache, and a defined cache size of 50,000 kilobyte blocks (the machine *must* have this many blocks available on its disk):

```
/...:dcelocal/var/adm/dfs/cache:50000
```

19.4 Altering Default Parameters with the dfsd Process

The fields in the **CacheInfo** file are the only Cache Manager parameters that you *must* set. However, you can use the options available with the **dfsd** command to alter several Cache Manager defaults, affecting the way information is cached. Following are the configuration parameters that have the greatest effect on cache performance. (See the following subsections for a description of the **dfsd** options used to configure these parameters.)

- Total Cache Size: This is the amount of disk space or memory available for caching.
- Chunk Size: This parameter determines the maximum amount of data that can fit in a cache chunk. A chunk cannot hold more than one element; in a memory cache, the unused memory that is allocated for a chunk is wasted. If an element cannot fit in a single chunk, it is split into as many chunks as are needed.

  This parameter also determines the maximum amount of data that the Cache Manager can request at one time from a File Exporter. Increase the chunk size to take advantage of very fast links or decrease the size for slow networks.

- Cache Chunk Configuration: This parameter determines the number of chunks that are used for the cache. It can affect how often the Cache Manager must discard cached data to make room for new data. Consider the following example:

  A disk cache is configured at 50 megabytes and consists of 1000 chunks. Suppose 10 users each have the Cache Manager cache 100 files and each file is 20 kilobytes in size. This uses all 1000 chunks available (because each chunk can hold only one element), even though the cache has only 20 megabytes of cached elements, which is less than 50% of its capacity of 50 megabytes.

  When a user requests more data, the Cache Manager must discard cached data to reclaim space, even though the cache is not close to its capacity. In this case, increasing the number of chunks into which the cache is divided would improve performance by allowing the unused 30 megabytes of cache capacity to be allocated for other cached files.

- Number of dcache Entries in Memory: The Cache Manager maintains one dcache entry for each cache chunk; the entry records system identification information such as the file ID and version number of the file corresponding to the chunk. On disk caching machines, each dcache entry is stored in the **CacheItems** file, with a small number of entries (by default, 100) duplicated in machine memory. On memory caching machines, all dcache entries are stored in memory; the number of entries is equal to the number of chunks.

Note: The **dfsd** command is normally placed in a machine's initialization file (**/etc/rc** or its equivalent), not typed at the console. The commands in the following subsections are presented as examples of entries from initialization files.

19.4.1 Disk Cache Configuration

The number of kilobyte blocks allocated to the cache is defined in the third field of the **CacheInfo** file. You can use the **-blocks** option to override the number of cache blocks; the unit of measure associated with the cache block size is always kilobytes. The Cache Manager heuristically divides the number of blocks by 8 to determine the number of cache chunks in a disk cache. The following example sets the number of disk blocks allocated for the cache to 75,000 kilobyte blocks:

```
dfsd -blocks 75000
```

The default number of cache chunks in a disk cache is computed as the number of cache blocks divided by 8; you can use the **-files** option with a positive integer not greater than 32,000 to override this default. To use your cache most effectively, issue the **du** command on the cache directory to determine the number of cache blocks used; compare this number to the number of blocks allocated to the cache. If you are not using 90% of the cache, increase the number of chunks (**V** files). The following example sets the number of chunks to 2000:

```
dfsd -files 2000
```

The default chunk size for a disk cache is 64 kilobytes (2^{16}); the unit of measure associated with the chunk size is always bytes. You can use the **-chunksize** option to override the default chunk size. Provide an integer between 13 and 18 to be used as an exponent of 2. For example, a value of 15 sets the chunk size to 32 kilobytes (2^{15}= 32,768); a value of 16 equals the default for disk caches (2^{16}= 64 kilobytes). A value less than 13 or greater than 18 returns the chunk size to the default (as does a value of 16). The following example sets the chunk size to 16 kilobytes (2^{14}):

```
dfsd -chunksize 14
```

For a disk cache, the default number of dcache entries duplicated in memory is 100. You can use the **-dcache** option with a positive integer to change the default. It is usually not necessary to duplicate more than 100 entries in memory. However, because memory access is faster than disk access, increasing the number of dcache entries stored in memory may improve performance slightly. The following example sets the number to 250:

```
dfsd -dcache 250
```

When altering a disk cache configuration, any combination of **dfsd** options is allowed. However, the cache size defined in the **CacheInfo** file or with the **-blocks** option cannot be exceeded with the **-files** or **-chunksize** option.

19.4.2 Memory Cache Configuration

The default chunk size for a memory cache is 8 kilobytes (2^{13}). There is no predefined default for the number of chunks in a memory cache, and, as mentioned previously, the number of dcache entries equals the number of chunks.

If the **-blocks** option is used alone, it overrides the default cache size in the **CacheInfo** file. The Cache Manager divides this value by the default chunk size of 8 kilobytes to calculate the number of chunks and dcache entries. The following example sets the cache size to 5 megabytes (5120 kilobytes); as a result, the number of chunks is set to 640 (5120 divided by 8):

```
dfsd -memcache -blocks 5120
```

If the **-chunksize** option is used alone, it overrides the default of 8 kilobytes (2^{13}). Provide an integer between 13 and 18 to be used as an exponent of 2. For example, a value of 15 sets the chunk size to 32 kilobytes (2^{15}= 32,768). A value less than 13 or greater than 18 returns the chunk size to the default (as does a value of 13). The following example sets the chunk size to 16 kilobytes (2^{14}); if the total cache size is 4 megabytes (2^{12} kilobytes), the resulting number of chunks is 256:

```
dfsd -memcache -chunksize 14
```

If the **-blocks** and **-chunksize** options are used together, they override the defaults for the cache size and the chunk size. The Cache Manager divides the cache size by the chunk size to calculate the number of chunks and dcache entries. The following example sets the cache size to 8 megabytes (8192 kilobytes) and the chunk size to 16 kilobytes (2^{14}), resulting in 512 chunks:

```
dfsd -memcache -blocks 8192 -chunksize 14
```

When configuring a memory cache, the following options explicitly set the number of chunks and dcache entries. They also set the cache size indirectly and should not be used; use **-blocks**, **-chunksize**, or both, and allow the Cache Manager to determine the number of chunks and dcache entries itself.

- The **-dcache** option alone. The Cache Manager multiplies this value by the default chunk size (8 kilobytes) to derive a total cache size, overriding the value in the **CacheInfo** file.
- A combination of **-dcache** and **-chunksize** options. The Cache Manager sets the specified values and multiplies them together to obtain a total cache size, again overriding the value in the **CacheInfo** file.

Do not use the following options when configuring a memory cache:

- The **-files** option alone. This option sets the number of chunks for a disk cache and is thus ignored for a memory cache.
- The **-blocks** and **-dcache** options together. If you combine these options, the Cache Manager may choose one of the values to ignore, or the command may fail.

(See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about these options and the use of **dfsd**.)

19.5 Changing the Cache Location

The default directory for the Cache Manager's cache is *dcelocal*/**var/adm/dfs/cache**. You can change this to a directory on another partition if more space is available. Use the **du** or **df** command (or an equivalent command) to check partition size and fullness.

You can change the location of the cache by editing the **CacheInfo** file or by using the **-cachedir** option with the **dfsd** command. (See the DCE DFS portion of the *OSF DCE Administration Reference* for more information about the use of the **dfsd** command.)

To change the cache location, do the following:

1. Log in as **root** on the machine.
2. Use a text editor to change the second field of the **CacheInfo** file (the information between the two colons). The new directory must be on the local disk of the machine. The following example shows the cache directory specified in the **CacheInfo** file changed to **/usr/cache**:

```
/...:/usr/cache:15000
```

3. Reboot the machine using **/etc/reboot** or its equivalent. (Consult your system documentation for information on the correct rebooting command for your workstation.)
4. Move to the old cache directory and delete it using the **rm -rf** command.

19.6 Listing and Setting Cache Size

The amount of local disk space or memory allocated for the Cache Manager to use for its cache affects the speed of file access. A larger cache means the Cache Manager has to contact the File Server machine less often, resulting in fewer cross-network messages. A smaller cache fills sooner, making it more likely that the Cache Manager must discard cached copies of data to make room for newly requested data; if the user requests the discarded data again, the Cache Manager must recontact the File Server machine and refetch the data. A larger cache can make the initial discarding of data unnecessary.

The amount of disk space or memory used for caching depends on several factors. The size of the partition that houses the cache directory or the amount of memory available on the machine places an absolute limit on cache size. Do not use more than 90% of the cache directory's partition for a disk cache; do not use more than 20 to 25% of available memory for a memory cache (this leaves enough memory for processes and applications to run).

Within these limits, devoting more than 40 megabytes to the cache on a machine that does not serve multiple users is normally not useful unless users often work with large amounts of data (accessing large databases, for example). If a machine serves multiple users, a cache of 60 to 70 megabytes may be appropriate. A cache smaller than 5 megabytes can hamper Cache Manager performance; a cache smaller than twice the chunk size is rounded up.

You can reset the cache size for both types of caches by using a text editor to alter the size field in the **CacheInfo** file and then rebooting the machine; you must be logged in as **root** or have the write permission on the file to edit it. The **-blocks** option can also be used with the **dfsd** command to override the **CacheInfo** value at reboot. (See the DCE DFS portion of the *OSF DCE Administration Reference* for complete information about the **dfsd** process.)

To alter disk cache size without rebooting the machine, use the **cm setcachesize** command. The value remains in effect until the machine is next rebooted and reads the value in the **CacheInfo** file. To display the current cache size, the amount being used, and the type of cache (disk or memory), use the **cm getcachesize** command.

You must reboot to reset the cache size for a memory-caching machine.

19.6.1 Displaying the Cache Size from the CacheInfo File

Use the **cat** command (or the command appropriate to your system) to view the **CacheInfo** file on a client machine:

```
$ cat CacheInfo
```

The **CacheInfo** file contains a single line that lists three fields separated by a colon; the third field lists the maximum number of kilobyte blocks the Cache Manager can reserve for use as a cache in the designated cache directory.

In the following example, the default cache size for the machine is 25,000 kilobyte blocks:

```
$ cat CacheInfo

/.../dcelocal/var/adm/dfs/cache:25000
```

19.6.2 Displaying the Current Cache Size and the Amount in Use

Issue the **cm getcachesize** command on a client machine to view the current size of the cache and the amount in use. On machines that use disk caching, the current cache size may disagree with the default size specified in the **CacheInfo** file if the cache size was changed with the **cm setcachesize** command. Regardless of the type of caching in use, the current cache size may also disagree with the **CacheInfo** file if the size was changed with the **-blocks** option of the **dfsd** command.

```
$ cm getcachesize
```

The following example shows the number of kilobyte blocks that the Cache Manager is using as a cache at the moment the command is issued and the current size of the cache:

```
$ cm getcachesize

Using 13709 of the cache's available 25000 1K byte blocks.
```

19.6.3 Changing the Cache Size Temporarily

You can reset the cache size without rebooting a machine. The value remains in effect until you next reboot the machine. To change the cache size temporarily, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setcachesize** command to set a new cache size:

   # **cm setcachesize -size** *kilobytes*

   The **-size** *kilobytes* option is the number of kilobyte blocks to be used for the cache. A value of 1024 equals 1 megabyte; the smallest allowable value is 1. A value less than 5120 (5 megabytes) can have a negative effect on Cache Manager performance; a value less than twice the chunk size is rounded up. A value of 0 (zero) resets the cache size to the amount specified in the **CacheInfo** file.

19.6.4 Resetting the Cache Size to the Default

To reset the cache size to the default, do the following:

1. Log in as **root** on the machine.
2. Use the **-reset** option with the **cm setcachesize** command to reset the cache size to the value specified at the last reboot, which is either the value in the **CacheInfo** file or the value set with the **-blocks** option of the **dfsd** command:

   # **cm setcachesize -reset**

   The **-reset** option resets the cache size to the value at the last reboot.

19.6.5 Changing the Cache Size Permanently

To change the cache size permanently, do the following:

1. Log in as **root** on the machine.
2. Use a text editor to change the number in the third field of the **CacheInfo** file. Specify a number in kilobyte blocks; 1024 kilobyte blocks equals 1 megabyte. Following is an example of the **CacheInfo** file:

   ```
   /...:dcelocal/var/adm/dfs/cache:25000
   ```

   Caution: Be precise when editing the **CacheInfo** file. Use colons to separate the fields in the file; do not include any spaces in the file.
3. Reboot using **/etc/reboot** or its equivalent. (Consult your system documentation for information on the correct rebooting command for your workstation.)

19.7 Determining setuid Permission

Programs that have **setuid** permission allow users to perform operations and access local files for which they normally may not have the necessary permissions. Such a program allows anyone who uses it to execute with the permissions of the user who owns the program for the duration of the program's execution.

While a **setuid** program executes, the person executing it is treated as the owner of the program. The effective **uid** of the executing program is the **uid** of the person who owns the program, not the **uid** of the person who initiated the program's execution. Thus, the person executing the program is granted the same permissions as the person who owns the program for as long as the program executes.

A **setuid** program owned by **root** allows a user who executes the program to execute with **root** privilege for the duration of the program. When handled correctly, such **setuid** programs are very useful. For example, programs that modify the password file for a system (**/etc/passwd** or its

equivalent) are **setuid** programs that allow users to execute with **root** privilege long enough to modify their passwords. When handled incorrectly, however, **setuid** programs owned by **root** can present a serious breach in security.

In the UNIX operating system, **setuid** programs are indicated by setting a mode bit associated with a file. By default, the Cache Manager does not allow **setuid** programs to execute with **setuid** permission. Use the **cm setsetuid** command to enable **setuid** programs from specific filesets to execute with **setuid** permission. The command sets **setuid** status on a per-fileset and per-Cache Manager basis. It is commonly included in a start-up file (**/etc/rc** or its equivalent) to enable **setuid** programs from a specified fileset at machine startup.

Note that **setuid** programs are effective only in the local environment. A **setuid** program can change only the local identity under which a program runs; it cannot change the DCE identity with which a program executes because it provides no Kerberos tickets. DCE does not recognize the change to the local identity associated with a **setuid** program.

Use the **cm getsetuid** command to determine whether the Cache Manager allows programs from specific filesets to execute with **setuid** permission.

Note: Every program also has a **setgid** bit that, when set, allows a person executing the program to execute with the permissions of the group that owns the program for the duration of its execution. When the **cm setsetuid** command is used, it automatically enables or disables **setgid** permission at the same time. Thus, if **setuid** programs are enabled on a fileset, **setgid** programs are also enabled on that same fileset.

19.7.1 Checking setuid Permission

Issue the **cm getsetuid** command to determine the status of **setuid** programs on specific filesets:

$ **cm getsetuid** [**-path** {*filename* | *directory_name*}...]

The **-path** option specifies a file or directory from each fileset whose **setuid** status is to be displayed. Omit this option to display the status for the fileset that contains the current working directory.

The output from this command includes a line for each specified fileset, stating either

- `no setuid allowed`, indicating that **setuid** (and **setgid**) programs from the fileset are disabled
- `setuid allowed`, indicating that **setuid** (and **setgid**) programs from the fileset are enabled
- `cm: the fileset on which 'pathname' resides does not exist`, indicating that the pathname specified with the **-path** option is invalid

19.7.2 Changing setuid Permission

To change **setuid** permission, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setsetuid** command to change the status of **setuid** (and **setgid**) programs on specific filesets:

   # **cm setsetuid** [**-path** {*filename* | *directory_name*}...] [**-state** {**on** | **off**}]

   The **-path** option specifies a file or directory from each fileset whose **setuid** status is to be changed. Omit this option to change the status for the fileset that contains the current working directory.

   The **-state on** option allows **setuid** programs from the indicated filesets to execute with **setuid** permission; the **-state off** option prevents **setuid** programs from the indicated filesets from executing with **setuid** permission. If this option is omitted, **setuid** programs from the specified filesets are allowed to execute with **setuid** permission.

19.8 Determining Device File Status

In UNIX file systems, devices are represented as special device files. By convention, device files reside in the **/dev** directory or a subdirectory of that directory; the UNIX kernel always honors device files stored in the **/dev** directory. However, the Cache Manager determines whether device files stored in filesets in the global namespace are honored. By default, the Cache Manager does not honor device files stored in filesets in the global namespace.

You can use the **cm setdevok** command to instruct the Cache Manager to honor device files stored on specific filesets. This is especially useful for diskless machines. Because the **/dev** directory cannot be a local directory, the Cache Manager must be instructed to honor device files stored in filesets used by diskless machines. The **cm setdevok** command sets device file status on a per-fileset and per-Cache Manager basis. The command is commonly included in a start-up file (**/etc/rc** or its equivalent) to honor device files at machine startup.

Use the **cm getdevok** command to determine whether the Cache Manager honors device files from specific filesets.

19.8.1 Checking Device File Status

Issue the **cm getdevok** command to determine whether the Cache Manager honors device files on specific filesets:

$ **cm getdevok** [**-path** {*filename* | *directory_name*}...]

The **-path** option specifies a file or directory from each fileset about which device file status information is to be displayed. Omit this option to display the status for the fileset containing the current working directory.

The output from this command includes one line for each specified fileset, stating either

- `device files allowed`, indicating that device files from the fileset are honored

- `device files not allowed`, indicating that device files from the fileset are not honored
- `cm: the fileset on which 'pathname' resides does not exist`, indicating that the pathname specified with the **-path** option is invalid

19.8.2 Changing Device File Status

To change device file status, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm setdevok** command to change the status of device files on specific filesets:

   # **cm setdevok [-path** {*filename* | *directory_name*}...] **[-state** {**on** | **off**}]

   The **-path** option specifies a file or directory from each fileset for which device file status is to be changed. Omit this option to change the status for the fileset containing the current working directory.

   The **-state on** option causes device files from the indicated filesets to be honored; the **-state off** option prevents device files from the indicated filesets from being honored. If this option is omitted, device files from the specified filesets are honored.

19.9 Updating Cached Data

When an application program requests new data and the cache is full, the Cache Manager discards some data to make room for the new information. It discards data based on the following two factors:

- If the data is reproducible; information is considered reproducible if it is unchanged from its first retrieval. By definition, data from read-only filesets is always reproducible; data from read/write filesets that was changed by a local application is considered reproducible if the changes are stored to the File Server machine.

- When the application program last referenced the data; data not used for the longest time is discarded first.

Thus, reproducible data not used for the longest time is discarded first. The Cache Manager continues to discard least recently used (LRU) data in this fashion until there is enough room for the new data.

You can force the Cache Manager to discard, or flush, data cached from files, directories, and filesets. You can flush individual files or directories with the **cm flush** command, or you can flush one or more filesets with the **cm flushfileset** command. Flushing is necessary only in the event of file system problems or for testing purposes. The **cm flush** and **cm flushfileset** commands do not cause the Cache Manager to discard changes to data not written back to the central copies of files. These commands also do not affect data in the buffers of application programs.

The Cache Manager checks once an hour for changes that do not involve tokens, such as the release of a new version of a cached read-only fileset or a name change for any cached fileset. You can force the Cache Manager to notice these changes at other times with the **cm checkfilesets** command, which directs the Cache Manager to revise its table of mappings between fileset names and fileset ID numbers.

19.9.1 Flushing Specific Files or Directories

Issue the **cm flush** command to discard data from specific files or directories:

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

The **-path** option names each file or directory that you want to flush from the cache. If a *directory_name* is used, the Cache Manager flushes the name mappings and the blocks associated only with the directory, not with the files in the directory. If this option is omitted, the current working directory is flushed.

19.9.2 Flushing All Data from Specific Filesets

Issue the **cm flushfileset** command to discard data from specific filesets:

$ **cm flushfileset** [**-path** {*filename* | *directory_name*}...]

The **-path** option names each file or directory in a fileset whose contents you want to flush. The Cache Manager flushes everything cached from each fileset that contains a specified file or directory. If this option is omitted, the fileset that contains the current working directory is flushed.

19.9.3 Forcing the Cache Manager to Notice Other Fileset Changes

Issue the **cm checkfilesets** command to make the Cache Manager check for changes to information about filesets that contain cached data:

$ **cm checkfilesets**

19.10 Discarding Unstored Data

The Cache Manager may occasionally be unable to write cached data back to a File Server machine, possibly because the File Server machine is down or because network problems prevent the Cache Manager from reaching it. In this event, the Cache Manager displays a message on the screen to notify the user that it cannot write the data to the File Server machine. If possible, it also returns a failure code to the application program that is using the data.

The Cache Manager keeps the unstored data in the cache and continues to attempt to contact the File Server machine until it can store the data. (The frequency with which the Cache Manager attempts to reach a File Server machine is defined with the **-pollinterval** option of the **fxd** command issued on that File Server machine.) In the meantime, corrective measures can be taken to alleviate the problem that prevents the data from being stored; for example, the File Server machine can be restarted. Once the problem is alleviated, the Cache Manager can contact the File Server machine and store the data.

The Cache Manager discards unstored data only when

- It needs to make room in the cache for other data. Given an average-sized cache with average usage, the Cache Manager rarely needs to discard unstored data.
- The **cm resetstores** command is issued to force the Cache Manager to discard unstored data from the cache. This command cancels the Cache Manager's continued attempts to contact unavailable File Server machines; *all* data that the Cache Manager cannot store to such File Server machines is discarded; you cannot selectively discard individual files or data from specific filesets.

The **cm resetstores** command affects only data that could not be written to a File Server machine; it does not affect other data in the cache. Nonetheless, issue the command only after issuing the **cm lsstores** command. The **cm lsstores** command lists the fileset ID numbers of filesets that contain data that the Cache Manager cannot write to a File Server machine. Examine the output of the command to be sure that you know from which filesets unstored data will be discarded. You may be able to use this information to ensure that unstored data from the indicated filesets can safely be discarded.

Note: Because unstored data discarded from the cache cannot be recovered, any problem that prevents data from being written to a File Server machine should be handled promptly.

19.10.1 Listing Unstored Data

Issue the **cm lsstores** command to list filesets that contain unstored data that the Cache Manager cannot write back to a File Server machine:

```
$ cm lsstores
```

19.10.2 Discarding Unstored Data

To discard unstored data, do the following:

1. Log in as **root** on the machine.
2. Issue the **cm resetstores** command to cancel any further attempts by the Cache Manager to contact unavailable File Server machines. The Cache Manager discards all data that it has been unable to store to such File Server machines.

   ```
   # cm resetstores
   ```

Chapter 20

Configuring the Backup System

The DFS Backup System can help you automate the process of making permanent copies of filesets on tape. You can create a full backup, which includes all of the data from every file in a fileset, or you can back up data incrementally, copying only those files that have changed since a previous dump. In the same fashion, you can restore filesets completely, or you can do an incremental, date-specific restore, which re-creates the filesets as they were before a specific date. Because both DCE LFS filesets and exported non-LFS file systems have entries in the Fileset Location Database (FLDB), the DFS Backup System can be used with both types of filesets.

This chapter introduces the DFS Backup System. It describes configuration issues related to the performance of backup and restore operations. Chapter 21 provides specific details about listing information from the Backup Database, backing up and restoring data, and administering the Backup Database. Refer to this chapter to configure the Backup System and to prepare it for backing up and restoring data; refer to Chapter 21 to back up and restore data.

20.1 Introduction to the Backup System

With the DFS Backup System, you control many aspects of the backup process, including how often backups are performed, which filesets are backed up, and whether full or incremental backups are made. A dump or dump set is the result of performing a backup operation; it includes data from all of the filesets that were copied onto tape at the same time. A full dump includes data from every file in a fileset; an incremental dump includes only those files in the fileset that changed since a previous dump was made. The backup process is also referred to as *dumping a fileset family* or *creating a dump set*.

Once a fileset has been dumped, the DFS Backup System can be used to restore it. When restoring a fileset, the DFS Backup System first restores the most-recent full dump of the fileset. It then restores the changes to the fileset from any incremental dumps that were made since the last full dump. Two types of restores are possible: a full restore, which re-creates the fileset as it was at its last dump, including changes from both the last full dump and any incremental dumps that were made since the full dump; and a date-specific restore, which re-creates the fileset as it was at the time of its last dump before an indicated date (the Backup System restores the changes from any incremental dumps that were done before the specified date, so the fileset is current according to that date).

The Backup System can be used to dump and restore data between different types of file systems. For example, data dumped from a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of non-LFS fileset. Likewise, data dumped from a non-LFS fileset can be restored to a DCE LFS fileset or to a different type of non-LFS fileset. Note that incompatible information may be lost when a fileset is dumped and restored between file system types; for example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs. (Refer to your vendor's documentation to verify the level of support for dump and restore operations between different types of file systems.)

The Backup Database records the schedule for backups, the locations of the Backup System's Tape Coordinators, the groups of filesets (fileset families) that can be dumped, and other administrative information. One Backup Database exists per cell; it is used to back up data from all administrative domains in the cell. A master copy of the Backup Database is maintained on one machine and replicated on other machines in the cell. Ubik creates and

synchronizes the master and secondary copies of the database. (See Chapter 13 for more information about Ubik.) In addition, the DFS Backup System provides facilities to back up the database by copying it to tape so that it can be restored if necessary. You can also remove specific configuration and dump information from the database if needed.

The Backup Database is maintained by the Backup Server, or **bakserver** process. The Backup Server must run on each machine that stores a copy of the Backup Database. Only the administrative users and members of the groups included in the **admin.bak** administrative list can issue **bak** commands, which are used to configure and administer the Backup System and to back up and restore data. Like the Backup Database, the **admin.bak** file is installed on one machine (usually the System Control machine) and copied to all machines that house copies of the Backup Database.

Some operating systems have their own backup commands. If your operating system has commands named **bak**, make certain that you use the complete pathname (*dceshared*/**bin/bak**) when issuing DFS **bak** commands. Note that **bak** commands can presently be used to affect the Backup Database in the local cell only.

20.1.1 Tape Coordinator Machines

A Tape Coordinator machine is a machine on which backup and restore operations are physically conducted. To qualify as a Tape Coordinator machine, a machine

- Should be in a physically secure location
- Must have one or more tape drives attached
- Must be configured as a Tape Coordinator machine
- Must run one instance of the **butc** (BackUp Tape Coordinator or just Tape Coordinator) process for each tape drive

The **butc** program must be active when you issue a **bak** command that involves either a tape drive or an operation being performed by a tape drive. For optimum efficiency, run several tape drives (and their Tape Coordinators). Start one **butc** process on a Tape Coordinator machine for each tape drive attached to the machine. Each Tape Coordinator controls the behavior of its associated drive and accepts service requests from the **bak** program.

A Tape Coordinator ID (TCID) identifies a Tape Coordinator; each TCID must be unique in the Backup System of the local cell. When you issue **bak** commands, specify a Tape Coordinator by specifying its TCID with the **-tcid** option. Depending on the command, the **bak** or **butc** program contacts one or more of the following: the Backup Database (by way of the Backup Server), the FLDB, or Fileset Server processes.

20.1.2 Fileset Families and Fileset Family Entries

When creating backups, you copy groups of filesets, known as fileset families, to tape. A fileset family includes all of the filesets that you want to dump together onto the same tape (or tapes, if the fileset family contains a large number of filesets). All of the filesets in a fileset family are dumped to tape with the same frequency (for example, once a day or once a week).

Fileset family entries (also referred to as fileset entries) define the filesets included in a fileset family. Each entry has three fields: the File Server machine name, the aggregate name, and the fileset name. Because filesets can be moved from File Server to File Server, the first two fields are usually designated with a .* wildcard, so a person backing up the filesets does not need to know the File Server machine and aggregate on which they reside. The last field, fileset name, is often specified with a regular expression pattern that matches certain fileset names.

With the Backup System, regular expression characters and the .* wildcard can be used in many arguments. (See Section 20.3.2 for a description of these characters and their interpretations.)

20.1.3 Dump Hierarchies and Dump Levels

A dump hierarchy is a logical structure that helps define the relationship between full and incremental dumps. As mentioned previously, an incremental dump includes only the files that changed since the fileset was last dumped; when creating an incremental dump, the Backup System uses a previous dump, known as the dump's parent, to serve as a reference point on which to base the incremental dump.

A dump set is the product of dumping a fileset family at a certain dump level, which is an entry in the dump hierarchy. The dump set is a collection of data from filesets dumped at the same time and in the same manner (fully or incrementally). To create a dump set, you specify (with the **bak dump** command) both the fileset family and the dump level. The Backup System keeps all of the data in a dump set together on a tape (or set of tapes, if the dump set is too large to fit on a single tape). The name of the dump set consists of the name of the fileset family and the last component of the name of the dump level joined by a period (*fileset_family_name.dump_level*).

Each dump level can be associated with an expiration date that specifies when a tape containing data from that dump level can be overwritten. The expiration date is transferred to any backup tape that contains a dump made at that level. When dumping to tape, the system checks the tape for an expiration date. If the tape's expiration date has not expired (if it is in the future), the system does not overwrite the tape; if no expiration date is defined for the tape or if the tape's expiration date has expired (if it is in the past), the system overwrites the tape, but only with a dump set of the same name.

20.1.4 Command and Monitoring Windows

A single terminal session can be used to issue **bak** commands to the Tape Coordinators on all Tape Coordinator machines. The session corresponds to a command shell called the command window, which can be opened and closed without affecting the Tape Coordinators. This session can be run from any machine in the cell. Multiple command windows can be used, but they are not necessary.

A separate terminal session must be used for each Tape Coordinator and associated tape drive running on a machine; a terminal session of this type is referred to as a Tape Coordinator's monitoring window. The window must be a connection to the Tape Coordinator machine whose Tape Coordinator and tape drive it is monitoring. The Tape Coordinator runs in the foreground, so no further commands can be issued in the monitoring window. The monitoring window must remain open while the Tape Coordinator runs so that you can see the Tape Coordinator's prompts.

20.1.5 Privileges Required to Use the Backup System

Three administrative lists, or **admin** files, determine the users who can perform specific backup and restore operations. Depending on the operation to be performed, you must be included in one or more of the following files:

- The **admin.bak** file on each server machine on which the Backup Database is stored. You must be listed in this file for any operation initiated by a **bak** command.
- The **admin.fl** file on each server machine on which the FLDB is stored. You must be included in this file for any operation that involves the Fileset Location Server (FL Server), such as restoring filesets to a File Server machine.
- The **admin.ft** file on any File Server machine from which you dump filesets or to which you restore filesets. You must be listed in this file for any command that involves the Fileset Server, such as backing up or restoring filesets.

To avoid confusion, include any user who works with the Backup System on all three administrative lists on the appropriate machines. The operations in this chapter direct users to verify that they, meaning they or a group to which they belong, are included in the appropriate administrative lists (**admin.bak**, **admin.fl**, and **admin.ft**). Use the **bos lsadmin** command to display the members of an administrative list.

Note: If you have not included users who are to issue **bak** commands on all three of these administrative lists, some commands may not work as detailed in this and the following chapter.

20.2 Standard Information in this Chapter

The following subsections present standard options and arguments common to many of the commands described in this chapter. They also present some common operations repeated throughout this chapter.

20.2.1 Standard Options and Arguments

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

- The **-tapehost** *machine* option is the DCE pathname of the machine (for example, **/.../abc.com/hosts/bak1**) for which a Tape Coordinator is to be added.
- The **-family** *fileset_family_name* option is the name of the fileset family to be used in the command. The name must be unique within the Backup Database of the local cell. It can be no longer than 31 characters. It can include any characters, but to avoid confusion when dump set names are created, it should not include a **.** (period). Any regular expression characters entered on the shell command line must be escaped with a \ (backslash); for example, **usr*** for a fileset family named **usr***. To make it easier to track the contents of a fileset family, its name should give some indication of the contents of the fileset entries it contains; for example, use the name **user** for the fileset family that includes all user filesets in the file system.
- The **-level** *dump_level* option is the name of the dump level to be used in the command. The complete pathname of a dump level must always be specified. There are two types of dump levels:
  - — Full dumps, which consist of a name preceded by a single / (slash); for example, **/full**.
  - — Incremental dumps, which consist of multiple elements that resemble a UNIX pathname listing the dump levels that serve as the parents of the dump level, starting with a full dump level and proceeding in order down the hierarchy; for example, **/full/weekly/monday**. An incremental dump level can consist of any number of elements; when defining a new dump level, all of the elements except the last one must already exist. Each level in a dump level name must be preceded by a / (slash).

  Dump levels should have meaningful names that give some indication of their purpose. A single element in a dump level name can be no longer than 28 characters, and the complete name can be no longer than 256 characters. Dump level names can include any characters, but to avoid

confusion when dump set names are created, they should not include a . (period). Regular expression characters included in a dump level name must be escaped with a \ (backslash) or " " (double quotes). The complete pathname of each dump level must be unique within the Backup Database of the local cell.

- The **-expires** *date* option is the expiration date for a dump level. Expiration dates can be specified in one of two ways:

  — Relative expiration dates, which use the keyword **in** to indicate a number of years, months, or days to be added to the current date to calculate the expiration date. When the system dumps a fileset at this level, it calculates the time at which the dump set expires by adding the values to the start time of the dump operation. Relative expiration dates are expressed as follows:

  in [*integer***y**] [*integer***m**] [*integer***d**]

  At least one value must be provided; multiple values must be listed in the order shown, with the appropriate unit abbreviation (**y**, **m**, or **d**) used with each value. For example, **in 1y 6m 2d** causes the system to add 1 year, 6 months, and 2 days to the current date to calculate the expiration date.

  — Absolute expiration dates, which use the keyword **at** to represent a specific date and, optionally, time to use as the expiration date. Absolute expiration dates are expressed as follows:

  at *mm/dd/yy* [*hh*:*mm*]

  If you specify a time, you must use 24-hour time. For example, **at 11/22/92 11:36** specifies an expiration date and time of 22 November 1992 at 11:36 a.m. If no time is provided, a default time of 00:00 (12:00 a.m.) on the indicated date is used.

If you omit the **-expires** option from a command, tapes created at dump levels specified with the command have no expiration dates; they can be overwritten at any time. Also, although the **-expires** options are followed by ellipses, you can specify only one expiration date. The ellipses are included only to accommodate the DFS command parser.

- The **-tcid** *tc_number* option is the TCID of the Tape Coordinator to be used for the command. Legal values are integers from 0 (zero) to 1023. If this option is omitted, the Tape Coordinator with a TCID of 0 is used to execute the command by default.

20.2.2 Standard Commands and Operations

The following subsections describe commands and operations that are used frequently in this chapter. If a command or operation is described in detail here, it generally is not described in depth in later sections of this chapter where it is used.

20.2.2.1 Starting a Tape Coordinator

Before performing a backup or restore operation, you must install at least one tape drive on a Tape Coordinator machine and define its corresponding Tape Coordinator in both the *dcelocal*/**var/dfs/backup/TapeConfig** file and the Backup Database by using the **bak addhost** command. (See Section 20.3 for a description of these operations.) This section explains how to start a Tape Coordinator. You must have a Tape Coordinator running any time you access a tape drive for use with the Backup System.

1. Make certain that you have the **w** (write) and **x** (execute) permissions on the *dcelocal*/**var/dfs/backup** directory, which is the directory in which the Tape Coordinator creates its **TL** (log) and **TE** (error) files.
2. Start a new terminal session on the Tape Coordinator machine to use as the monitoring window for the Tape Coordinator. It must remain open while the Tape Coordinator runs.
3. In the newly opened window, issue the **butc** command to start the Tape Coordinator. The binary file for the **butc** program resides in the *dceshared*/**bin** directory.

   ```
   $ butc [-tcid tc_number] [-debuglevel trace_level]
   ```

The **-debuglevel** *trace_level* option specifies the type of messages to be displayed. There are two valid arguments:

1 Indicates that the Tape Coordinator is to report on its activities as it restores filesets, in addition to prompting for new tapes as necessary.

0 Indicates that the Tape Coordinator is only to prompt for new tapes; it also displays some output as necessary for operations that it executes. This is the default.

20.2.2.2 Stopping a Tape Coordinator

When you are finished using a Tape Coordinator, you should stop it from running. To stop a Tape Coordinator process, enter an interrupt signal (**<Ctrl-c>** or its equivalent) in the Tape Coordinator's monitoring window.

20.2.2.3 Determining Tape Size and End-of-File Mark Size

The size of a tape determines the amount of data the Backup System can place on it. The tape size differs for different tape drives. In addition, the Backup System appends an end-of-file (EOF) mark after each fileset it dumps to tape. The size of the mark also affects the amount of space available for backup data on a tape. The values used for both of these figures are specified in the **TapeConfig** file once for each tape drive. Note that an EOF mark is appended after each fileset, not after each file.

If you do not know the tape capacity or EOF mark size for a tape drive, use the **fms** (file mark size) command to determine these values. The binary file for this command resides in the *dceshared*/**bin** directory. This command produces terminal output and an **FMSLog** file in the current directory; both the output and the **FMSLog** file list the tape capacity and the size of the EOF mark for the drive.

Note: Because this command inserts file marks onto the entire tape, it can take from several hours to more than a day to complete.

To determine the EOF mark size for a tape drive, do the following:

1. Make certain you have the **w** (write), **x** (execute), and **i** (insert) ACL permissions on the directory from which the command is issued. If the **FMSLog** file already exists in the directory, you need to have only the **w** permission on the file.
2. Insert a tape into the tape drive. The tape is overwritten while the command executes; you may want to use a blank tape or one that can be recycled.
3. Enter the **fms** command:

   $ **fms -device** *device_name*

   The **-device** *device_name* option specifies the name of the tape drive.

   An example of this command and its terminal output follows; the command also writes similar information to the **FMSLog** file. In the example, the tape size for the drive named **/dev/rmt1h** is 151,224,320 bytes; the EOF mark size for the drive is 2,375,680 bytes.

   $ **fms /dev/rmt1h**

```
wrote block: 9320
Finished data capacity test - rewinding
wrote 9230 blocks, 9230 file marks
Finished file mark test
Tape capacity is 151224320 bytes
File marks are 2375680 bytes
```

20.2.2.4 Using the Interactive Interface

You can use the **bak** commands in regular command mode or in interactive mode. If you use interactive mode, note the following:

- You do not need to enter the string **bak** with each **bak** command; the bak> prompt replaces the command shell prompt.
- You do not have to escape regular expression characters; in regular command mode, you must place all regular expressions and wildcards in " " (double quotes) or escape each with a \ (backslash).

- You can track executing and pending operations with the **bak jobs** command; in regular command mode, you cannot track operations.
- You can cancel currently executing and pending operations with the **bak kill** command; in regular command mode, you cannot use the **bak kill** command.
- You do not have to establish a new connection each time you issue a command, so execution time is quicker; in regular command mode, each command establishes new connections to the **bakserver** and **flserver** processes, as necessary.

Most of the operations described in this chapter are presented in regular command mode. Where appropriate, some operations include steps introduced as "*Optional*" to indicate where working in interactive mode could be useful. The **bak jobs** and **bak kill** commands can be entered *only* in interactive mode.

20.2.2.4.1 Entering Interactive Mode

Enter the **bak** command:

```
$ bak
```

20.2.2.4.2 Leaving Interactive Mode

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

```
bak> quit
```

20.3 Configuring the Backup System

Before using the Backup System for backing up and restoring data, you must ensure that certain conditions have been met. The following subsections explain in detail how to perform the following prerequisite tasks:

- Configuring Tape Coordinator machines
- Defining fileset families and fileset family entries
- Defining a dump hierarchy of dump levels
- Labeling tapes, if necessary

See Chapter 21 for information on inspecting the status of these prerequisites. If all of the prerequisites are met, turn to the appropriate section of Chapter 21 for information on using the Backup System.

20.3.1 Configuring a Tape Coordinator Machine

Setting up a Tape Coordinator machine consists of using **bak** commands to configure the Tape Coordinators for the machine. You must also create the **TapeConfig** file, which includes a line for each Tape Coordinator on the machine. Each line defines the

- Size of tapes used in the drive, in kilobyte, megabyte, or gigabyte units.
- End-of-file (EOF) mark size for the drive. EOF marks are placed between filesets on a tape. The size of the EOF mark can differ for each type of tape drive.
- Device name (for example, **/dev/rst0**) of the drive.
- Tape Coordinator ID (TCID) of the drive.

Each tape drive and its Tape Coordinator must be assigned a TCID in the range 0 to 1023; the TCID must be unique in the Backup Database of the local cell. When assigning the TCID, you must define the ID number in the Backup Database with the **bak addhost** command; you must also edit the **TapeConfig** file on the local disk of the Tape Coordinator machine.

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

1. Prepare the tape drives.
2. Create the **TapeConfig** file that defines the tape parameters for each drive.
3. Create an entry in the Backup Database for the Tape Coordinator for each drive.

The information that follows details the order in which you need to perform all of the required tasks:

1. Install one or more drives on the machine according to the manufacturer's instructions. The Backup System can track a maximum of 1024 drives in a cell.
2. If you are configuring a client machine as a Tape Coordinator machine, verify that the *dcelocal*/**var/dfs** and *dcelocal*/**var/dfs/backup** directories exist on the machine. Create the directories if they do not already exist.
3. Verify that you have the **w** (write) and **x** (execute) permissions on the *dcelocal*/**var/dfs/backup** directory (the directory in which you must create the **TapeConfig** file).
4. Create the *dcelocal*/**var/dfs/backup/TapeConfig** file on the machine with a text editor. Use a single line in the file for each tape drive attached to the Tape Coordinator machine, recording the following information:
   - The tape size of the tapes to be used in the drive. The Tape Coordinator uses this capacity as the size of all tapes used in the drive. It is recommended that you use a number 10 to 15% lower than the actual tape capacity to allow for tape variations. The following abbreviations can be used for the tape size unit of measurement (the default is kilobytes); do not leave a space between the number and the letter.
     — Kilobytes: k or K (for example, 2k or 2K)
     — Megabytes: m or M (for example, 2m or 2M)
     — Gigabytes: g or G (for example, 2g or 2G)

- The EOF mark size for the type of tape to be used in the drive. The Backup System appends an EOF mark after each fileset dumped to tape. The size of this mark can affect the amount of space available for backup data. The EOF mark size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. It is recommended that you increase the actual file mark size by 10 to 15% to allow for tape variations.

  If you do not specify a unit of measurement, the default unit used for the EOF size is bytes (*not* kilobytes, as for tape capacity). To indicate other units, use the same abbreviations as for tape capacity.

- The device name of the tape drive. The format of this name varies with each operating system. For example, in the UNIX operating system, a valid device name is **/dev/rst0**.

- The TCID for the Tape Coordinator associated with the drive. The Backup System can track a maximum of 1024 tape drives; legal values are integers from 0 to 1023. You do not have to assign the numbers in sequence, and you can skip numbers. The TCID for any Tape Coordinator must be unique among all TCIDs in the local cell.

  Because the **bak** commands that require you to specify a TCID always use a default TCID of 0, assign a TCID of 0 to the Tape Coordinator for the drive that you will use most often; this enables you to omit the **-tcid** option as often as possible.

If you do not know the tape size or the EOF mark size for the tape drive, determine them by using the **fms** command, as described in Section 20.2.2.3.

Following is an example of the contents of the **TapeConfig** file for a machine with two drives. The tape size for each drive is 2 gigabytes; the EOF mark size for each drive is 1 megabyte. The respective TCIDs of the two drives are 0 and 1.

```
2g 1m /dev/rmth0h 0
2G 1M /dev/rmth1h 1
```

5. Ensure that the **butc** binary file is stored in the *dceshared*/**bin** directory of the machine. If you are configuring a client machine as a Tape Coordinator machine, the file should already be present as a

result of installation; if you are not using a client machine or if the file is not present, create the *dceshared*/**bin** directory on the local disk of the machine, if necessary, and copy the file from the source tape.

6. Verify that the individuals who are to use the Backup System are included in the appropriate administrative lists; if necessary, issue the **bos lsadmin** command to check. You also need to ensure that you are included in the **admin.bak** list to issue the **bak addhost** command that follows. To add someone to a list, issue the **bos addadmin** command.
7. Verify that the **bakserver** process is running on the cell's Backup Database machines. If necessary, issue the **bos status** command to check.
8. *Optional*. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. The advantages of interactive mode are described in Section 20.2.2.4. The command in the following step assumes that regular command mode is used, *not* interactive mode.
9. Enter the **bak addhost** command to create an entry in the Backup Database for each Tape Coordinator, defining its TCID:

   $ **bak addhost -tapehost** *machine* [**-tcid** *tc_number*]

   Repeat the **bak addhost** command for each Tape Coordinator.

20.3.2 Defining Fileset Families and Fileset Family Entries

Before actually performing a backup, you must create one or more fileset families in the Backup Database. A fileset family defines the groups of filesets that are to be dumped together. Fileset family names can be no longer than 31 characters, and they can include any characters. Avoid using a period in the name of a fileset family; when a dump set is transferred to tape, the fileset family name and the last component of the dump level name are automatically joined by a period to form the name of the dump set.

In regular command (noninteractive) mode, characters from the regular expression character set used in the name of a fileset family must be escaped with a \ (backslash) to prevent the command shell from expanding

them; for example, **usr*** for a fileset family named **usr***. Because they have no meaning in the name of a fileset family, regular expression characters are not recommended.

Once you define a fileset family, you must then define the fileset family entries in the family. Each fileset family entry is defined in terms of one or more filesets and the location of each fileset on a File Server machine and aggregate. Each fileset family entry consists of three fields, with each field separated by a space. Following are the legal values for each field in a fileset family entry:

- File Server Machine Name: The DCE pathname of the File Server machine (for example, **/.../abc.com/hosts/fs1**) that houses the filesets. You can use the special **.*** wildcard for this field; this wildcard matches all of the File Server machines in the cell.
- Aggregate Name: The device name or aggregate name of the aggregate on which the filesets reside. You can use the **.*** wildcard for this field; the wildcard matches all aggregate names.
- Fileset Name: The names of the filesets to be backed up. You can use the **.*** wildcard for this field to match all fileset names. The following regular expression characters can also be used in this field of an entry:
  - — The * (asterisk) character matches any number of repetitions of the previous character.
  - — The [ ] (brackets) characters around a list of characters match any single instance of the characters in the list but no other characters.
  - — The ^ (circumflex) character as the first character in a bracketed set of characters matches any single character other than the characters that follow it in the list.
  - — The **?** (question mark) character matches any single character or no character.
  - — The **.** (period) character matches any single character, but a character must be present.
  - — The \ (backslash) character before any other regular expression character, including itself, matches the literal value of the character.

  In noninteractive mode, you must surround an entire string with " " (double quotes) if it contains regular expression characters or you must escape each regular expression character with a \ (backslash); for example, **"user\..*\.bak"** or **user\\..*\\.bak** to indicate all of the

filesets that begin with the prefix **user.** and end with the extension **.bak**. Otherwise, the command shell attempts to resolve the regular expression characters rather than pass them to the **bak** command interpreter for resolution. Note that the .* notation is interpreted as a single wildcard that must be surrounded with double quotes in noninteractive mode ("**.***"). Characters specified in regular expressions are case sensitive.

All fileset family names must be unique within the Backup Database of the local cell. Create and delete fileset families with the **bak addftfamily** and **bak rmftfamily** commands. Create and delete fileset family entries with the **bak addftentry** and **bak rmftentry** commands.

20.3.2.1 Suggestions for Creating Fileset Family Entries

The **bak addftentry** command has arguments that correspond to the three fields in a fileset family entry: **-server** for the File Server machine name field, **-aggregate** for the aggregate name field, and **-fileset** for the fileset name field. By combining these arguments in different ways, you can create fileset entries for different groupings of filesets. Table 20-1 summarizes some suggested groupings.

Table 20–1. Suggestions for Creating Fileset Family Entries

For Entries That Include:	Use:	For Example:
All filesets in the cell's file system	The wildcard for all three arguments	".*" ".*" ".*"
Every fileset on a specific File Server machine	The machine name with **-server** and the wildcard for **-aggregate** and **-fileset**	**/.../abc.com/hosts/fs1** ".*" ".*"
Filesets on aggregates of the same name	The aggregate name with **-aggregate** and the wildcard for **-server** and **-fileset**	".*" **/dev/lv01** ".*"
Every fileset with a common string of letters (such as a **.backup** extension)	The wildcard for **-server** and **-aggregate**, and a character string/regular expression for **-fileset**	".*" ".*" **".*\.backup"**
All filesets on an aggregate	The machine name with **-server**, the aggregate name with **-aggregate**, and the wildcard for **-fileset**	**/.../abc.com/hosts/fs2** **/dev/lv02** ".*"
Every fileset on each File Server machine's similarly named aggregate that includes a common string of letters in its name (such as a **user.** prefix)	The wildcard for **-server**, the aggregate name with **-aggregate**, and a character string/regular expression for **-fileset**	".*" **/dev/lv03** **"user\..*"**
Every fileset on one aggregate with a common string of letters in its name (such as a **sys** prefix and a **.readonly** extension)	The machine name with **-server**, the aggregate name with **-aggregate**, and a character string/regular expression for **-fileset**	**/.../abc.com/hosts/fs3** **/dev/lv04** **"sys.*\.readonly"**

Include in a fileset family only those filesets that you wish to dump to the same tape at the same time (for example, weekly or daily) and in the same manner (fully or incrementally).

The two main types of fileset families are those based on a common fileset location (File Server machine and aggregate) and those based on similar contents (as reflected by a fileset name). Because filesets can be moved between machines and aggregates, use name-based fileset family entries rather than location-based ones. For name-based entries, specify the **.*** wildcard for the **-server** and **-aggregate** arguments of the **bak addftentry** command.

20.3.2.2 Adding a Fileset Family to the Backup Database

To add a fileset family to the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak addftfamily** command to create the fileset family. The fileset family remains empty until you use the **bak addftentry** command to define entries in it.

   $ **bak addftfamily -family** *fileset_family_name*

20.3.2.3 Adding a Fileset Family Entry to a Fileset Family

To add a fileset family entry to a fileset family, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Define the entries in a fileset family that was previously created with the **bak addftfamily** command; the Backup System automatically assigns each entry an index number, starting with 1 for the first entry in each fileset family. This number is used if the fileset entry needs to be removed.

   $ **bak addftentry -family** *fileset_family_name* **-server** *machine*
   -aggregate *name* **-fileset** *name*

The **-server** *machine* option is the DCE pathname of the File Server machine (for example, **/.../abc.com/hosts/fs1**) that houses the fileset to be included in the entry. The **.*** wildcard can be used to match the names of all of the File Server machines in the cell.

The **-aggregate** *name* option is the device name or aggregate name of the aggregate that houses the fileset to be included in the entry. The **.*** wildcard can be used to match the name of any aggregate.

The **-fileset** *name* option is the name of the fileset to be included in the entry. The **.*** wildcard or any of the regular expression characters described previously can be used to match the names of multiple filesets.

20.3.2.4 Deleting Fileset Families from the Backup Database

To delete a fileset family from the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak rmftfamily** command to delete each fileset family that you no longer need. It is not necessary to delete the fileset entries in each fileset family first; they are deleted automatically when the family is removed.

   ```
   $ bak rmftfamily -family fileset_family_name...
   ```

20.3.2.5 Deleting a Fileset Family Entry from a Fileset Family

To delete a fileset family entry from a fileset family, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak lsftfamilies** command to determine the index number of the entry that you want to delete. This is necessary only if the fileset family contains multiple entries, in which case this command is used to list the entries; if the family contains a single entry, the index is 1.

$ **bak lsftfamilies -family** *fileset_family_name*

3. Use the **bak rmftentry** command to delete the entry:

   $ **bak rmftentry -family** *fileset_family_name* **-entry** *fileset_entry_index*

   The **-entry** *fileset_entry_index* option is the index for the entry that you want to delete.

20.3.3 Defining a Dump Hierarchy of Dump Levels

A dump hierarchy consists of one or more full dump levels and any incremental dump levels that you create with the **bak adddump** command. The dump levels define how fileset families are to be dumped; all fileset family entries in a fileset family are dumped at the same time and in the same way (fully or incrementally). A dump of a fileset family at a particular dump level produces a dump set. To create a dump set, specify the name of the fileset family and the level at which that family is to be dumped when you initiate the dump with the **bak dump** command. (See Chapter 21 for a description of the **bak dump** command.)

A dump hierarchy is defined by the dump levels it contains. The term *full dump level* refers to a dump level used when creating full dumps; the term *incremental dump level* refers to a dump level used when creating incremental dumps; the term *parent dump level* refers to a dump level that serves as the reference point for an incremental dump level. Both full dump levels and incremental dump levels can serve as parent dump levels.

Each dump level in the hierarchy can be associated with an expiration date that specifies the date and time at which a tape that contains a dump set made at that level can be overwritten. Expiration dates are specified with the **bak adddump** or **bak setexp** command. A dump level's expiration date is automatically placed on a tape that contains a dump made at that level to provide an extra level of protection against accidental erasure of the information on the tape.

Whenever a tape is used, the Backup System always checks to see whether the tape already contains a dump set. If the tape contains a dump set, the Backup System overwrites the tape only with a dump set of the same name. If the Backup System determines that it can overwrite the dump set, it then determines whether an expiration date exists on the tape; if no expiration

date is associated with a tape or if the expiration date associated with a tape has expired, the system overwrites the dump set on the tape with a dump set of the same name. However, if the tape's expiration date has not expired, the system refuses to overwrite the tape.

Following are some general issues to consider when building a dump hierarchy:

- A dump level can have any number of elements. The / (slash) is used as a metacharacter to separate different levels in the dump hierarchy. Regardless of its level in the dump hierarchy (full or incremental), each element in a dump level name must be preceded by a / (slash).
- Any characters can be included in a dump level name. Regular expression characters included in a name must be properly escaped with a \ (backslash) or " " (double quotes).
- Do not include a . (period) in the name of a dump level. When a dump set is transferred to tape, the last component of the dump level name becomes part of the dump set name. The elements of the dump set name (the fileset family name and the last component of the dump level name) are joined by a period. For example, if a fileset family named **sys** is dumped at the incremental dump level **/weekly/monday**, the dump set name is **sys.monday**.
- The maximum length for any single element in a dump level name is 28 characters. This does not include the / (slash) that precedes the element.
- The maximum length for the complete name of a dump level (full or incremental) is 256 characters. This includes any / (slashes) that are part of the name.
- A dump level is specified by its pathname. A level can share parents, but the level itself must have a unique name. Following are examples of different dump specifications:
  - The **/full** specification defines a full dump level.
  - The **/full/week1** specification defines an incremental dump level, **/week1**, with **/full** as its parent.
  - The **/full/week1/thursday** specification defines **/thursday** as a dump level that refers to **/week1** as its parent; **/week1** refers to **/full** as its parent.
- The dump level that you use as the parent for an incremental dump must already exist in the hierarchy when you define the incremental dump.

The complete pathname of each dump level must be unique within the Backup Database of the local cell.

- A dump hierarchy can contain more than one full dump level; each level defines a separate subhierarchy in which you can create different relationships between the dump levels. The following two common methods are available to relate the incremental dumps in a subhierarchy to the full dump level and to one another:

  — Each incremental dump refers to the same full dump as its parent. With this method, the dump sets created at each of the incremental levels contain all of the files in the fileset family that changed since the family was last dumped at the full level.

  — Each incremental dump level (other than the first) refers to a preceding incremental dump level as its parent, rather than to the full dump level. With this method, each incremental dump includes only those files modified since a dump was last done at its parent level. When you restore files dumped in this fashion, however, you must access more tapes: the tape that contains the full dump and the tape for each incremental dump done afterward.

  The two types of hierarchies can be mixed within a single subhierarchy by setting some incremental dumps to refer to the full dump level as their parent and setting others to refer to preceding incremental levels.

- There is no implied relationship between a fileset family and a dump subhierarchy; you can dump any fileset family at any level in any subhierarchy. When dumping a fileset, *do not* alternate between incremental dumps from different subhierarchies. To dump a fileset according to a different subhierarchy, start at the full dump level.

- Use names in the hierarchy that correspond to real-world times; these can help you remember when to create dumps at the different levels. However, the Backup System does not automatically back up filesets according to the names assigned in the dump hierarchy; it does not interpret dump level names, nor does it automatically perform an incremental dump on Thursday simply because there is a dump level called **thursday**.

A few general guidelines for using a dump hierarchy follow:

- To set up a group of tapes for archiving, make certain that you use unique dump names; for example, **monday1**, **tuesday1**, **monday2**, or **tuesday2**.

- To recycle tapes, use dump levels with the same name; for example, **monday** or **tuesday**.
- To archive tapes and recycle them at a later time, simply perform backups with a new set of tapes; the old set of tapes can then be archived. This creates multiple entries for the dump in the Backup Database. To restore filesets with multiple entries in the database, use the correct dump date to restore the correct information.

The Backup System prevents you from using out-of-date configuration information. For example, if a user deletes a full dump level in a hierarchy, and another user tries to start an incremental backup based on that full dump level, the incremental backup fails. The second user must view the dump hierarchy with the **bak lsdumps** command. This command updates the hierarchy with the most recent changes and lets the user determine another dump level to use with the command. (See Chapter 21 for more information on the **bak lsdumps** command.)

20.3.3.1 Examples of Dump Hierarchies

Following are examples of two possible dump hierarchies. Each hierarchy backs up data in a different way.

The first dump hierarchy is used to back up user data (data from user filesets). Because this data changes frequently, it is dumped at the end of each working day, starting with a full dump at the beginning of the week (Sunday) and continuing with incremental dumps Monday through Friday. Each incremental dump refers to the full dump as its parent, rather than to the previous day's incremental dump. As a result, each incremental dump contains all of the data that has changed since the full dump was performed. The following commands are used to establish this dump hierarchy:

```
$ bak adddump /sunday
$ bak adddump /sunday/monday
$ bak adddump /sunday/tuesday
$ bak adddump /sunday/wednesday
$ bak adddump /sunday/thursday
$ bak adddump /sunday/friday
```

You can use the **bak lsdumps** command to display the dump hierarchy. In the following example, **/sunday** is the full dump used for the hierarchy:

```
$ bak lsdumps

/sunday
        /monday
        /tuesday
        /wednesday
        /thursday
        /friday
```

The second dump hierarchy is used to back up filesets containing system binary files. Because these files do not change often, they are backed up only once a week, starting with a full dump at the beginning of the month and followed by an incremental dump at the beginning of each subsequent week. Each weekly dump refers to the previous week's dump as its parent, rather than to the initial full dump. Therefore, each weekly dump contains only those files that changed in the past week, rather than everything that changed since the full dump was performed. The following commands establish this dump hierarchy:

```
$ bak adddump /month
$ bak adddump /month/week1
$ bak adddump /month/week1/week2
$ bak adddump /month/week1/week2/week3
$ bak adddump /month/week1/week2/week3/week4
```

You can use the **bak lsdumps** command to display the dump hierarchy. The following example shows **/month** as the full dump for the hierarchy:

```
$ bak lsdumps

/month
        /week1
                /week2
                        /week3
                                /week4
```

20.3.3.2 Defining a Dump Level

To define a dump level, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak adddump** command to define one or more dump levels:

   $ **bak adddump -level** *dump_level...* [**-expires** *date...*]

20.3.3.3 Changing a Dump Level's Expiration Date

To change a dump level's expiration date, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak setexp** command to set the expiration dates of one or more dump levels:

   $ **bak setexp -level** *dump_level...* **-expires** *date...*

20.3.3.4 Deleting a Dump Level

To delete a dump level, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Issue the **bak rmdump** command to delete a dump level. All dump levels for which the level serves as the parent, either directly or indirectly, are also deleted automatically.

   $ **bak rmdump -level** *dump_level*

20.3.4 Labeling Tapes

A tape's magnetic label provides information about the tape and the data it contains. The Backup System checks each tape before it writes to it; if the label on the tape is unacceptable, the dump cannot proceed until you insert an acceptable tape in the drive. A tape's label records the following information:

- The name of the tape, indicating its contents. The name is composed of three fields that are separated by periods: *fileset_family_name.dump_level.index* (the dump set name with an additional tape index). The following three types of tape names are acceptable:
  - — The complete name in the form *fileset_family_name.dump_level.index*, where the *fileset_family_name* and the *dump_level* match values that you provide with the **bak dump** command. The *index* is this tape's place in the sequence of tapes used for the complete dump set; if the dump set fits on one tape, the index for that tape is the numeral 1.
  - — An indicator of empty, or null, created with the **bak labeltape** command. The Backup System replaces the null indicator with the correct name when it puts dump sets onto the tape.
  - — No name, indicating it is an unused tape. Again, the Backup System generates the correct name as it transfers a dump set to the tape.
- The size of the tape. Use a number and a letter (k or K for kilobytes, m or M for megabytes, or g or G for gigabytes) to indicate a size, as described in Section 20.3.1 for the **TapeConfig** file. Because the Backup System always uses the size specified in the **TapeConfig** file, the size you include in the label of the tape is intended for information purposes only.

When you label a tape, you can specify its name only, its size only, or both its name and its size. When you dump data to a tape, the expiration date of the dump level at which you dump the data is copied to the label of the tape. If a tape has an expiration date that has not expired, the Backup System refuses to overwrite the tape. If the tape's expiration date has expired, or if the tape contains no expiration date, the Backup System overwrites the tape with a dump set that has an acceptable name.

It is not essential to prelabel tapes before data is transferred to them; the Backup System can use unlabeled tapes or partially labeled tapes, which are tapes that include only the name of the tape or the size of the tape. However, you may want to prelabel a tape if:

- You want to automate the backup process as much as possible. If tapes are prelabeled with the correct name, the individual performing backups needs only to respond to prompts when the system requests a tape.
- You wish to reuse a tape, putting a different dump set on it. The Backup System will not use a tape if the label reflects the name of a different dump set or if the label contains an unexpired expiration date. Labeling a tape overwrites its expiration date, as well as its name and size.

20.3.4.1 Reading the Label on a Tape

To read the label on a tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 20.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. Place the tape in the drive, and issue the **bak readlabel** command to read the label on the tape.

   $ **bak readlabel** [**-tcid** *tc_number*]

20.3.4.2 Labeling a Tape

To label a tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 20.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)

2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. Issue the **bak labeltape** command to label the tape. This command executes in the background. (See Chapter 21 for more information about commands that execute in the background.)

   $ **bak labeltape** [**-tape** *tape_name*] [**-size** *tape_size*] [**-tcid** *tc_number*]

   The **-tape** *tape_name* option is the name of the tape. It must have the form *fileset_family_name.dump_level.index*. If this option is omitted, the tape is marked as empty with the null indicator.

   The **-size** *tape_size* option is the capacity of the tape. The default unit is kilobytes. You can add a g or G to the number to indicate gigabytes or an m or M to indicate megabytes. This is for information purposes only; the Backup System uses the tape size recorded in the **TapeConfig** file whenever it uses a tape drive. If this option is omitted, the tape size specified for the drive in the **TapeConfig** file is used.
4. Place the tape in the drive, and press **<Return>** in the corresponding Tape Coordinator's monitoring window.

20.4 Adding and Removing Tape Coordinators

As mentioned in Section 20.3.1, a separate Tape Coordinator process is associated with each tape drive on a Tape Coordinator machine. Each Tape Coordinator has an associated port, or address. You must assign the port a TCID and use that number in any commands issued to the Tape Coordinator. Any **bak** commands that involve a tape drive have a **-tcid** option for that purpose.

The Backup System can track a maximum of 1024 tape drives. Valid TCIDs are the integers from 0 to 1023. You do not have to assign the numbers in sequence, and you can skip numbers. The drive with the TCID of 0 is used by default.

Enter the **bak addhost** command to add an entry for a Tape Coordinator to the Backup Database. Enter the **bak rmhost** command to remove an entry

for a Tape Coordinator from the Backup Database. Use the **bak lshosts** command to list the Tape Coordinators that have entries in the Backup Database.

Remember to edit the **TapeConfig** file accordingly when you add or remove a Tape Coordinator. The **TapeConfig** file defines the mapping between a Tape Coordinator and a tape drive on a Tape Coordinator machine.

20.4.1 Adding a Tape Coordinator

To add a Tape Coordinator, do the following:

1. Install the drive on the machine according to the manufacturer's instructions.
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. Verify that you have the **w** (write) permission on the *dcelocal*/**var/dfs/backup/TapeConfig** file.
4. Choose the TCID for the drive. Enter the **bak lshosts** command to check previously assigned TCIDs:

   ```
   $ bak lshosts
   ```

5. Using a text editor, add a line for the new tape drive to the *dcelocal*/**var/dfs/backup/TapeConfig** file. Use a single line in the file for each tape drive, recording the following information:
   - The tape size of the tapes to be used in the drive. The Tape Coordinator uses this capacity as the size of all tapes used in the drive. It is recommended that you use a number 10 to 15% lower than the actual tape capacity to allow for tape variations. The following abbreviations can be used for the tape size unit of measurement (the default is kilobytes); do not leave a space between the number and the letter.
     — Kilobytes: k or K (for example, 2k or 2K)
     — Megabytes: m or M (for example, 2m or 2M)
     — Gigabytes: g or G (for example, 2g or 2G)

- The EOF mark size for the type of tape to be used in the drive. The Backup System appends an EOF mark after each fileset dumped to tape. The size of this mark can affect the amount of space available for backup data. The EOF mark size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. It is recommended that you increase the actual file mark size by 10 to 15% to allow for tape variations.

  If you do not specify a unit of measurement, the default used for the EOF size is bytes (*not* kilobytes, as for tape capacity). To indicate other units, use the same abbreviations as for tape capacity.

- The device name of the tape drive. The format of this name varies with each operating system.

- The TCID for the Tape Coordinator associated with the drive. Legal values are integers from 0 to 1023.

If you do not know the tape size or EOF mark size for the tape drive, determine them by using the **fms** command described in Section 20.2.2.3.

6. Enter the **bak addhost** command to define an entry in the Backup Database for the Tape Coordinator:
   $ **bak addhost -tapehost** *machine* [**-tcid** *tc_number*]

20.4.2 Removing a Tape Coordinator

To remove a Tape Coordinator, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. Verify that you have the **w** (write) permission on the *dcelocal*/**var/dfs/backup/TapeConfig** file.
3. Using a text editor, remove the line that defines the Tape Coordinator from the *dcelocal*/**var/dfs/backup/TapeConfig** file.
4. Enter the **bak rmhost** command to delete the entry for the Tape Coordinator from the Backup Database:

   $ **bak rmhost** [**-tcid** *tc_number*]

Chapter 21

Backing Up and Restoring Data

Once the DFS Backup System is properly configured, it can be used to help automate the process of making backup copies of both DCE LFS and non-LFS filesets on tape. These copies can then be used to restore data to the file system in the event of data loss. The Backup Database, which stores information about the dump schedule for backups, the locations of the Backup System's Tape Coordinators, the fileset families and their entries that can be dumped, and other administrative information, can also be backed up to tape and restored if the file system becomes damaged or corrupted.

This chapter describes how to use the Backup System to list backup information, back up file system data, and restore data to the file system if necessary. It also details the operations involved in administering the Backup Database, and describes canceling operations from the interactive interface.

The operations in this chapter assume that the Tape Coordinator machines are properly configured, the fileset families and fileset entries are defined, the dump hierarchy is defined, and the required tapes are labeled, as necessary. Chapter 20 described configuration of the Backup System in detail; make sure the Backup System is properly configured according to the guidelines detailed in Chapter 20 before attempting to use it to perform any of the operations described in this chapter.

21.1 Introduction to the Backup Process

Backing up, or dumping, data is the most basic operation performed with the Backup System. Data must be dumped to tape before it can be tracked in the Backup Database and before it can be restored from tape to the file system. This section provides an overview of the backup process.

Dumping a fileset makes it inaccessible to other file system users for the duration of the dump process. To reduce inconvenience, create a backup version of a fileset (a version with a **.backup** extension) and dump the backup version rather than the read/write version; this does not interrupt a user's work. Creating a backup version of a fileset, using the **fts clone** or **fts clonesys** command described in detail in Chapter 17, does make its read/write source fileset unavailable for a short period of time; therefore, you may wish to create backup versions during periods of low system usage, using **bos** commands to create a **cron** process to automate the procedure. (See Chapter 16 for a description of the **bos** commands.)

Occasionally, the Backup System cannot access a fileset, perhaps because of a File Server machine or Fileset Server outage. When this happens, it attempts to access the fileset three times over the course of the operation. If it still cannot access the fileset after the third attempt, it omits the fileset from the dump rather than aborting the dump or waiting for the fileset to become accessible. If the access failure occurs during a full dump, the next incremental dump of the fileset includes the entire fileset; if the failure occurs during an incremental dump, the next incremental dump of the fileset includes all files modified since the last successful dump of the fileset. You can set the Tape Coordinator that is performing the dump to notify you of the omission in its monitoring window (by specifying a value of **1** with the **-debuglevel** option of the **butc** command used to start the Tape Coordinator). The Tape Coordinator's error file also records the fileset's omission.

Following is a summary of the process the system uses to perform a typical backup. The example assumes that a backup is being performed on a Wednesday; the fileset family **usersys** is to be dumped at the dump level whose name in the dump hierarchy is **/sunday/wednesday** in this example. Note that the Backup System makes no implied connection between the name of a dump level and the date and time at which a dump at that level is to occur; descriptive dump level names serve merely as reminders to system administrators of when dumps are to be performed.

- The Backup System reads the dump hierarchy in the Backup Database to see if **/wednesday** is an incremental dump and, if so, to determine which preceding level is its parent. In this example, the **/wednesday** level is incremental, and **/sunday** is its parent.

```
/sunday
      /monday
      /tuesday
      /wednesday
      /thursday
      /friday
```

  If **/sunday** were specified, this would be a full dump; the system would copy the complete contents of each fileset in **usersys**. Because **/sunday/wednesday** is an incremental dump level, the dump set includes only those files that changed since **usersys** was dumped at the **/sunday** level.

- Because **/sunday** is the parent for **/wednesday**, the Backup System checks the Backup Database for the date and time of the last dump of **usersys** at the **sunday** level.

- The Backup System reads the fileset family **usersys** in the Backup Database to learn which fileset family entries it contains. The fileset family and its entries were created beforehand with the **bak addftfamily** and **bak addftentry** commands. In this example, the entries are

```
.* .* user.*
.* .* sys.*
```

- The Backup System scans the FLDB to match the wildcards from each fileset entry and generates a complete list of the filesets to be included in the dump. If duplicates are found, they are not dumped; only one occurrence of any fileset is included.

- The Backup System reads the label on the tape in the drive to verify that the tape name is acceptable and that the tape does not contain an unexpired expiration date.

- The system transfers the list of filesets to be backed up to the appropriate Fileset Server processes, which determine which data in the filesets was modified after the date and time of the last dump at the **/sunday** level.

- The designated Tape Coordinator puts the gathered data onto tape; the expiration date and other information associated with the dump are stored in the tape's label, and a unique dump ID number is assigned to the dump. If one tape is not large enough to hold the entire dump set, the Backup System prompts the operator to place additional tapes in the drive, as needed.

21.2 Standard Information in this Chapter

The following subsections present standard options and arguments common to many of the commands described in this chapter. They also present some common operations that are repeated throughout this chapter.

21.2.1 Standard Options and Arguments

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

- The **-family** *fileset_family_name* option is the name of the fileset family to be used in the command. To make it easier to track the contents of a fileset family, its name should give some indication of the contents of the fileset entries it contains (for example, **user** for the fileset family that includes all user filesets in the file system).
- The **-level** *dump_level* option is the name of the dump level to be used in the command. The complete pathname of a dump level must always be specified. There are two types of dump levels:
  - — Full dumps, which consist of a name preceded by a single / (slash); for example, **/full**.
  - — Incremental dumps, which consist of multiple elements that resemble a pathname listing the dump levels that serve as the parents of the dump level, starting with a full dump level and proceeding in order down the hierarchy; for example, **/full/weekly/monday**.

- The **-tcid** *tc_number* option is the TCID of the Tape Coordinator to be used for the command. Legal values are integers from 0 (zero) to 1023. If this option is omitted, the Tape Coordinator with a TCID of 0 is used to execute the command by default.

21.2.2 Standard Commands and Operations

The following subsections describe commands and operations that are used frequently in this chapter. If a command or operation is described in detail here, it generally is not described in depth in later sections of this chapter where it is used.

21.2.2.1 Starting a Tape Coordinator

Before performing a backup or restore operation, you must install at least one tape drive on a Tape Coordinator machine and define its Tape Coordinator in both the *dcelocal*/**var/dfs/backup/TapeConfig** file and the Backup Database, as described in Chapter 20. This section explains how to start a Tape Coordinator. You must have a Tape Coordinator running whenever you access a tape drive for use with the Backup System.

1. Make certain that you have the **w** (write) and **x** (execute) permissions on the *dcelocal*/**var/dfs/backup** directory, which is the directory in which the Tape Coordinator creates its **TL** (log) and **TE** (error) files.
2. Start a new terminal session on the Tape Coordinator machine to use as the monitoring window for the Tape Coordinator. It must remain open while the Tape Coordinator runs.
3. In the newly opened window, issue the **butc** command to start the Tape Coordinator. The binary file for the **butc** program resides in the *dceshared*/**bin** directory.

   ```
   $ butc [-tcid tc_number] [-debuglevel trace_level]
   ```

The **-debuglevel** *trace_level* option specifies the type of messages to be displayed. There are two valid arguments:

1 Indicates that the Tape Coordinator is to report on its activities as it restores filesets, in addition to prompting for new tapes as necessary.

0 Indicates that the Tape Coordinator only prompts for new tapes; it also displays some output as necessary for operations that it executes. This is the default.

21.2.2.2 Stopping a Tape Coordinator

When you are finished using a Tape Coordinator, you should stop it from running. To stop a Tape Coordinator process, enter an interrupt signal (**<Ctrl-c>** or its equivalent) in the Tape Coordinator's monitoring window.

21.2.2.3 Using the Interactive Interface

You can use the **bak** commands in regular command mode or in interactive mode. If you use interactive mode, note the following:

- You do not need to enter the string **bak** with each **bak** command; the bak> prompt replaces the command shell prompt.
- You do not have to escape regular expression characters; in regular command mode, you must place all regular expression characters in " " (double quotes) or escape each with a \ (backslash).
- You can track executing and pending operations with the **bak jobs** command; in regular command mode, you cannot track operations.
- You can cancel currently executing and pending operations with the **bak kill** command; in regular command mode, you cannot use the **bak kill** command.
- You do not have to establish a new connection each time you issue a command, so execution time is quicker; in regular command mode, each command establishes new connections to the **bakserver** and **flserver** processes, as necessary.

Most of the operations described in this chapter are presented in regular command mode. Where appropriate, some operations include steps introduced as "*Optional*" to indicate where working in interactive mode could be useful. The **bak jobs** and **bak kill** commands can be entered *only* in interactive mode.

21.2.2.3.1 Entering Interactive Mode

Enter the **bak** command:

```
$ bak
```

21.2.2.3.2 Leaving Interactive Mode

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

```
bak> quit
```

21.2.2.4 Using Commands That Execute in the Background

The following commands used with the Backup System execute in the background:

- **bak dump**
- **bak labeltape**
- **bak restoredb**
- **bak restoredisk**
- **bak restoreft**
- **bak savedb**
- **bak scantape**

As soon as you enter a command that executes in the background, the prompt at which you entered the command returns to the screen. The command continues to execute, but you can enter additional commands at the prompt while it executes. When you enter a command that does not execute in the background, the prompt does not return until the command is finished executing.

(See the sections in this chapter and in Chapter 20 for more information on these commands.)

21.2.2.5 Checking the Status of a Background Operation

You can check the status of a command that is executing in the background by

- Looking in the monitoring window for output from the command
- Entering the **bak status** command
- Entering the **bak jobs** command if you are working in interactive mode

Issue the **bak status** command to check the status of the operation that a Tape Coordinator is currently executing:

```
$ bak status [-tcid number]
```

The command produces output that includes the following:

- A name describing the operation the Tape Coordinator is performing. One of the following operation names is displayed:
  - `Dump` (*dump_set*) for a dump operation initiated with the **bak dump** command; *dump_set* is the name of the dump set in the form *fileset_family_name.dump_level*.
  - `Restore` for a restore operation initiated with the **bak restoreft** or **bak restoredisk** command.
  - `Labeltape` (*tape_label*) for a tape labeling operation started with the **bak labeltape** command; *tape_label* is the label being placed on the tape.
  - `Scantape` for a tape scanning operation initiated with the **bak scantape** command.

  - — `SaveDb` for a database saving operation initiated with the **bak savedb** command.
  - — `RestoreDb` for a database restoring operation started with the **bak restoredb** command.
- The number of kilobytes transferred so far (from the file system to tape for a dump operation, or from tape to the file system for a restore operation).
- For a dump operation, the string `fileset` followed by the name of the fileset currently being dumped; for a restore operation, the string `fileset` followed by the name of the fileset currently being restored.
- A message reporting additional status information about the operation. No message is displayed if the operation is proceeding normally.
  - — The `[abort requested]` message is displayed if the **kill** command was issued but the operation is not yet canceled.
  - — The `[abort sent]` message is displayed if the operation is canceled but its execution is not yet stopped.
  - — The `[operator wait]` message is displayed if the Tape Coordinator is waiting for the operator to insert a tape in the drive.

The following example shows the status of the operation currently being performed by the Tape Coordinator whose TCID is 0:

```
$ bak status

Dump (usersys.monday): 312105 Kbytes transferred, fileset user.terry.
```

(See Section 21.7.1 for information about the **bak jobs** command.)

21.3 Listing Backup Information

The following commands can be used to list information about the Backup Database:

bak verifydb	Checks the status of the Backup Database
bak lsftfamilies	Lists fileset families and fileset family entries
bak lsdumps	Lists the entries in the dump hierarchy
bak dumpinfo	Displays information about specific backups
bak lshosts	Lists Tape Coordinator IDs
bak ftinfo	Displays the dump history for a fileset

In addition to the preceding commands, which display information from the Backup Database, the **bak scantape** command reads a tape, extracting its tape label and information from the fileset header of each fileset on the tape. This command can detect damage to a tape that makes filesets incomplete; if it encounters damage, the scan aborts. All of the commands in the following subsections, like all **bak** commands, require that you be included in the **admin.bak** administrative list.

21.3.1 Verifying Backup Database Status

Issue the **bak verifydb** command to check the status of the Backup Database:

```
$ bak verifydb [-verbose]
```

The **-verbose** option directs the command to display additional information about the Backup Database.

Following is an example of the output from this command when the database is undamaged:

```
$ bak verifydb

Database OK.
```

21.3.2 Listing Fileset Families and Fileset Family Entries

Issue the **bak lsftfamilies** command to view a fileset family and its entries:

$ **bak lsftfamilies** [**-family** *fileset_family_name*]

The **-family** *fileset_family_name* option is the name of the fileset family whose entries are to be listed. Omit this option to view all of the fileset families and entries defined in the Backup Database.

Following is an example of the output from this command:

$ **bak lsftfamilies usersys**

```
Fileset family usersys:
 Entry 1: server .*, aggregate .*, filesets: user.*
 Entry 2: server .*, aggregate .*, filesets: sys.*
```

21.3.3 Listing Entries in the Dump Hierarchy

Issue the **bak lsdumps** command to view the entries in the dump hierarchy:

$ **bak lsdumps**

The following example shows a dump hierarchy. The **sunday** entry is the full dump level; the remainder of the entries are incremental dump levels that each have **/sunday** as their parent dump level.

$ **bak lsdumps**

```
/sunday
        /monday
        /tuesday
        /wednesday
        /thursday
        /friday
```

21.3.4 Viewing Recent Backup Information

Issue the **bak dumpinfo** command to list information about specific backups:

$ **bak dumpinfo** [{**-ndumps** *number* | **-id** *dumpID*}] [**-verbose**]

The **-ndumps** *number* option specifies the number of dumps about which information is to be displayed; information about the most recent number of dumps specified with this option is displayed. Use this option or use the **-id** option; omit both options to list information about the last 10 dumps.

The **-id** *dumpID* option specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this option or use the **-ndumps** option; omit both options to list information about the last 10 dumps.

The **-verbose** option includes a detailed list of information about the dump specified with the **-id** option. This option can be used only with the **-id** option.

The dump ID, parent ID, dump level, and other dump information are displayed about the indicated dumps. The following example shows information about the last three dumps:

$ **bak dumpinfo -ndumps 3**

```
    DumpID   parentID lvl      created      nt nfsets dump_name
-------------------------------------------------------------
 729293644  729289323   1  02/09/93  5:34  1     43 users.tue
 729287531  729286818   1  02/08/93  4:52  1     23 users.mon
 729286056          0   0  02/07/93  4:27  1     31 users.wkl
```

21.3.5 Listing Tape Coordinator TCIDs

Issue the **bak lshosts** command to list all of the Tape Coordinators that have entries in the Backup Database:

```
$ bak lshosts
```

The output lists the name of the machine for which each Tape Coordinator is defined and the TCID of the Tape Coordinator. A Tape Coordinator's presence in the output does not imply that it is currently running.

```
$ bak lshosts

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

21.3.6 Displaying a Fileset's Dump History

Issue the **bak ftinfo** command to display the dump history of a fileset:

```
$ bak ftinfo -fileset name
```

The **-fileset** *name* option names the fileset whose dump history is to be displayed. Include the **.backup** extension if the backup version of the fileset was dumped.

The dump ID, parent ID, dump level, and other dump information are displayed about dumps of the indicated fileset. The following example shows part of the dump history of the **user.smith.backup** fileset:

```
$ bak ftinfo user.smith.backup
```

```
  DumpID    parentID lvl creation date    clone date     tape name
654972910  654946323 1  10/01/91  5:07  10/01/91 4:01 users.tuesday.1
654960415  654946323 1  09/30/91  5:11  09/30/91 4:16 users.monday.1
654946323          0 0  09/29/91  5:36  09/28/91 4:31 users.week.1
```

21.3.7 Scanning the Contents of a Dump Tape

To scan the contents of a dump tape, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
2. Issue the **bak scantape** command to display information from a tape:

   $ **bak scantape [-dbadd]** [**-tcid** *tc_number*]

   The **-dbadd** option indicates that information extracted from the tape is to be added to the Backup Database; information is not added if the tape is damaged or if the entry has a dump ID number that is already used by an entry in the Backup Database. (See Section 21.6.3 for information about using this option.)
3. Place the tape in the drive, and press **<Return>** in the corresponding Tape Coordinator's monitoring window. When using this command, you must insert tapes sequentially.

   An example of the output from this command, which lists tape label and fileset information, follows. The output is displayed in the monitoring window of the Tape Coordinator:

   $ **bak scantape**

```
Tape label
----------
name =          guests.monthly.1
createTime =    Fri Nov 22 05:59:31 1990
cell =          /.../abc.com
size =          20103324 Kbytes
dump path =     /monthly
dump id =       729369701
```

```
useCount =      1
-- End of tape label --

-- fileset --
fileset name: user.guest10.backup
fileset ID 0,,112262
dumpSetName: guests.monthly
dumpID 729369701
level 0
parentID 0
endTime 0
clonedate Fri Nov 22 05:36:29 1991
```

21.4 Backing Up Data

The **bak dump** command is used to perform a dump operation. When you enter the command, specify the fileset family to be dumped and the level at which the family is to be dumped. All entries in the specified fileset family are dumped according to the dump level that you specify. If you specify a full dump level, all of the data in all of the filesets included in the specified family is dumped; if you specify an incremental dump, only the data in the filesets that has changed since the filesets were dumped at the parent of the dump level that you specify is dumped.

The fileset family and dump level that you specify produce a dump set. To indicate the contents of the dump set, the dump set name consists of the fileset family name joined by a period to the last component of the name of the dump level at which the family was dumped. For example, if the fileset family named **usersys** is dumped at the **/weekly/monday** level, the name of the resulting dump set is **usersys.monday**.

The Backup System overwrites a tape that contains an existing dump set only if both of the following conditions are true:

- The tape's label contains an acceptable name. An acceptable name is one that matches the name of the dump set you want to dump to the tape, in the form *fileset_family_name.dump_level.index*. Note that a tape's index is its position in the sequence of tapes necessary to accommodate the dump set; for example, the first tape for a dump set has an index of 1. A tape that is labeled as empty or that has no label is also acceptable.

- The tape's expiration date, if it exists, has expired. The Backup System refuses to overwrite a tape whose expiration date has not expired. Once a tape's expiration date has expired, the Backup System overwrites the contents of the tape with a dump set that has an acceptable name.

Use the **bak labeltape** command to overwrite a label that has an unacceptable name or an unexpired expiration date. Overwriting a tape's label removes all obstacles that can prevent it from being overwritten.

Note: If a dump operation is interrupted or fails for any reason, you cannot be sure that any fileset is complete on one tape; the Backup Database contains an entry for the incomplete dump set, which cannot be used to restore data. Immediately restart the backup, using the same tape to record the dump set; using the same tape automatically removes the entry for the incomplete dump set from the Backup Database. If you use a different tape, you will need to use the **bak deletedump** command to manually remove the entry for the incomplete dump set from the database. (See Section 21.4.3 for more information about the **bak deletedump** command.)

21.4.1 Using Tapes with a Backup Operation

You must place all dumps of a given fileset family (both full and incremental) onto tapes that are readable by a single tape drive. This is required because a single Tape Coordinator performs an entire restore operation, using a full dump set and any incremental dump sets as necessary. If a single Tape Coordinator cannot read all of the tapes on which the dump sets are recorded, you cannot restore all of the dumps of the fileset family.

For example, suppose the full dump of a fileset family is stored on 8-mm tape and the incremental dumps, which are done at a different time, are stored on streaming cartridge tape. When you restore a fileset from that fileset family, you must use a Tape Coordinator that uses 8-mm tapes because a restore always begins with the full dump. However, you cannot restore any of the incremental dumps because the same Tape Coordinator cannot read the streaming cartridge tapes and you cannot switch to another Tape Coordinator to continue the restore operation.

Before performing a backup, make sure the tapes are at least as large as the tape size listed in the **TapeConfig** file for the tape drive to be used for the operation. The Backup System fills a tape only with the amount of data listed as the capacity for the drive in the **TapeConfig** file. If a tape is larger than the tape size listed in that file, it simply is not filled to capacity when the backup is performed. However, if the tape is smaller than the size listed in the **TapeConfig** file, the backup operation fails, but only after it fills the tape and determines that it is too small for the drive.

A dump set does not have to fit entirely on a single tape; if the Backup System reaches the end of a tape while dumping a fileset from a fileset family, it puts the remaining data on another tape. The Backup Database automatically records that the fileset resides on multiple tapes.

Prior to performing a backup, you can preview the effects of your command without having the system actually perform the dump. Simply include the **-noaction** option with the **bak dump** command, specifying the remaining options as you would to really execute the dump. This lets you check a fileset family's size before actually dumping it so that you can calculate the correct number of tapes needed.

21.4.2 Backing Up a Fileset (Creating a Dump Set)

To back up a fileset, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
3. *Optional.* At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See Section 21.2.2.3 for the advantages of interactive mode.) The commands in the following steps assume that regular command mode is used, *not* interactive mode.
4. Decide which fileset family and dump level to use. If necessary, use the **bak lsftfamilies** or **bak lsdumps** command to display information about existing fileset families and dump hierarchies.

5. Check that you have a sufficient number of tapes; if you do not have enough tapes, you will not be able to complete the backup. Also, check that the tapes are *not* smaller than the tape size listed in the **TapeConfig** file for the drive. You must also check that the tapes are properly prelabeled (if necessary, use the **bak readlabel** command to check the labels); you must relabel any tape that

   - Is labeled with an incorrect name; tape names have the following format:

     fileset_family_name.dump_level.index.

   - Has an unexpired date; if a tape has an expiration date associated with it from a previous dump, you will not be able to use the tape unless the date is expired.

   If a label is incorrect, use the **bak labeltape** command to label the tape correctly.

6. Issue the **bak dump** command to dump the fileset family onto tape:

   $ **bak dump -family** *fileset_family_name* **-level***dump_level*
   [**-tcid** *tc_number*] [**-noaction**]

   The **-noaction** option specifies that all filesets that would be included in the indicated dump be displayed without the dump actually being performed. Specify all other options as you would to actually perform the operation.

7. Place the correct tape in the drive; the backup process begins immediately. If more than one tape is required, you must remain at the console to respond to prompts for subsequent tapes; if you do not respond immediately, a bell rings periodically to draw your attention.

21.4.3 Deleting Backup Information

The Backup System automatically removes the record of a dump set from the Backup Database when the tape containing the dump set is overwritten. The **bak deletedump** command can be used to manually remove information about a dump set from the Backup Database. It can be used to remove the record of a dump set that contains incorrect information (possibly because a dump operation was interrupted or failed) or for which

the corresponding tape is to be discarded. Before issuing the **bak deletedump** command, use the **bak dumpinfo** command to display the current dump IDs from the database.

After you use the **bak deletedump** command to delete the record of a dump set from the Backup Database, any dumps for which it serves as the parent, either directly or indirectly, are unusable. You can reissue the **bak deletedump** command to delete those dump sets from the database. However, leaving them in the database, while possibly confusing, causes no problems. Also, as long as the tape that contains the parent dump set remains available, you can always use the **bak scantape** command to restore dump set information about the parent from that tape to the database, making the dump sets that rely on the parent dump set usable again. (See Section 21.6.3 for more information about the **bak scantape** command.)

To delete backup information from the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. *Optional.* At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See Section 21.2.2.3 for the advantages of interactive mode.) The commands in the following steps assume that regular command mode is used, *not* interactive mode.
3. Issue the **bak dumpinfo** command to list information, including dump IDs, about dump sets recorded in the Backup Database. A dump set's dump ID is required to delete it from the Backup Database.

   $ **bak dumpinfo** [{**-ndumps** *number* | **-id** *dumpID*}] [**-verbose**]

   The **-ndumps** *number* option specifies the number of dumps about which information is to be displayed; information about the most recent **-ndumps** is displayed. Use this option or use the **-id** option; omit both options to list information about the last 10 dumps.

   The **-id** *dumpID* option specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this option or use the **-ndumps** option; omit both options to list information about the last 10 dumps.

   The **-verbose** option includes a detailed list of information about the dump specified with the **-id** option. The **-verbose** option can be used only with the **-id** option.

4. Issue the **bak deletedump** command to delete the desired dump set:

   $ **bak deletedump -id** *dumpID*

   The **-id** *dumpID* option specifies the dump ID number of the dump set whose entry is to be deleted from the Backup Database.

21.5 Restoring Data

When you restore data to the file system, you can restore individual filesets with the **bak restoreft** command, or you can restore an entire aggregate with the **bak restoredisk** command. Using the **bak restoreft** command, you can perform a full restore of the most recently dumped version of a fileset, or you can perform a date-specific restore, which restores the fileset to the state it was in at its last dump before the indicated date.

Data can be dumped and restored between different types of file systems. For example, data dumped from a DCE LFS fileset can be restored to a DCE LFS fileset or to any type of non-LFS fileset. Similarly, data dumped from a non-LFS fileset can be restored to a DCE LFS fileset or to a different type of non-LFS fileset. (See your vendor's documentation to verify the level of support for dump and restore operations between different types of file systems.)

Restored data is translated into the appropriate format for the file system to which it is restored. Note that incompatible information may be lost when a fileset is dumped and restored between different types of file systems. For example, ACLs on objects in a DCE LFS fileset may be lost if the fileset is restored to a file system that does not support ACLs.

Before performing a restore, you can direct the program to list the tapes needed to complete the operation. This allows you to locate and assemble the proper tapes before actually issuing the command. To view the list, include the **-noaction** option with the **bak restoredisk** or **bak restoreft** command, along with any other options that you intend to use with the command.

Note: If a restore operation is interrupted or fails for any reason, you cannot be sure that any fileset is complete in the file system; immediately restart the operation. If you do not, the file system may contain inconsistent information, which can result in problems in the future.

21.5.1 Specifying the Type and Destination of a Restore Operation

The system performs a full restore by default, re-creating a fileset as it existed when it was last dumped, including data from the last full dump and any subsequent incremental dumps. You can direct the system to perform a date-specific restore by using the **-date** option with the **bak restoreft** command. This re-creates the fileset as it was when it was last dumped before the indicated date; it includes the last full dump and any incremental dumps done before the indicated date.

Using either the **bak restoreft** or **bak restoredisk** command, you can restore data to the same location that it previously occupied, in which case it overwrites the existing information, or you can restore it with a new name or at a location other than its original storage location. To overwrite the current contents of a fileset, use the **-server**, **-aggregate**, and **-fileset** options with the **bak restoreft** command.

To preserve the current contents of a fileset in the file system, use the **-extension** option to restore the data to a new fileset with the same name as the existing fileset by adding a distinguishing extension, such as **.restored**, with the **-extension** option. A new FLDB entry is created for the fileset, and it is assigned its own fileset ID number. You can place the restored fileset at the same site as the existing fileset or at a different site by using the **-server** and **-aggregate** options.

To restore a fileset that no longer exists in the file system, simply provide the **-server**, **-aggregate**, and **-fileset** options. A new FLDB entry is created for the fileset and a fileset ID number is assigned.

Table 21-1 summarizes the options available with the **bak restoreft** command. Unless indicated as Optional in the table, each option is required.

Table 21–1. Options Available with the bak restoreft Command

Option	Specifies	Additional Information
-server	The File Server machine to which to restore each fileset	The specified machine can be a fileset's current site or a different site.
-aggregate	The aggregate to which to restore each fileset	The specified aggregate can be a fileset's current site or a different site.
-fileset	Each fileset to be restored	If you dumped the **.backup** version of a fileset, add the **.backup** extension to the name you specify.
-extension (Optional)	An extension to add to the name of each restored fileset	Specify an extension to preserve filesets in the file system that have the same names as those to be restored. If you want a period to separate the extension from each name, specify the period as the first character of the extension (for example, **.restored**).
-date (Optional)	The date and, optionally, the time to use for a date-specific restore	Only dumps sets of the indicated filesets dated before the specified date are restored. Omit this option to perform a full restore of the most recently dumped version of each fileset. Specify *mm/dd/yy* to indicate 00:00 (12:00 a.m.) on day *mm/dd/yy*; specify *mm/dd/yy hh:mm* to indicate time *hh:mm* on day *mm/dd/yy*. A time must be in 24-hour format (for example, **20:30** for 8:30 p.m.).

21.5.2 Restoring Individual Filesets

To restore a fileset, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. *Optional*. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See Section 21.2.2.3 for the advantages of interactive mode.) The command in the following step assumes that regular command mode is used, *not* interactive mode.
4. Issue the **bak restoreft** command with the appropriate options, using the information in Table 21-1 as a guide:

   $ **bak restoreft -server** *machine* **-aggregate** *name* **-fileset** *name...*
   [**-extension** *name_extension*] [**-date** *date*] [**-tcid** *tc_number*] [**-noaction**]

   The **-server** *machine* option is the DCE pathname of the File Server machine (for example, **/.../abc.com/hosts/fs1**) to which each fileset is to be restored.

   The **-aggregate** *name* option is the device name or aggregate name of the aggregate to which each fileset is to be restored.

   The **-fileset** *name* option is the name of each fileset to be restored.

   The **-extension** *name_extension* option is the new extension to be added to each fileset when it is restored.

   The **-date** *date* option specifies the date and, optionally, the time to use for a date-specific restore; only dumps performed prior to the specified date (and time) are included in the restore. There are three valid entries:

   0 Causes a full restore of the most recent dumps; this is the default.

   mm/dd/yy Causes a date-specific restore of dumps that were done before 00:00 (12:00 a.m.) on the indicated date.

mm/dd/yy hh:mm

Causes a date-specific restore of dumps that were done before the specified time on the indicated date. The time must be in 24-hour format; for example, **20:30** for 8:30 p.m. Surround the entire argument with " " (double quotes) because it contains a space.

The **-noaction** option specifies that the command is to display the list of tapes needed to complete the restore without performing the actual operation.

21.5.3 Restoring an Aggregate

To restore an aggregate, do the following:

1. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
2. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
3. *Optional*. At this point, you can issue the **bak** command at the system prompt to enter interactive mode. (See Section 21.2.2.3 for the advantages of interactive mode.) The command in the following step assumes that regular command mode is used, *not* interactive mode.
4. Issue the **bak restoredisk** command with the appropriate options:

   $ **bak restoredisk -server** *machine* **-aggregate** *name* [**-tcid** *tc_number*] [**-newserver** *machine*] [**-newaggregate** *name*] [**-noaction**]

   The **-server** *machine* option is the DCE pathname of the File Server machine (for example, **/.../abc.com/hosts/fs1**) that houses the aggregate to be restored.

   The **-aggregate** *name* option is the device name or aggregate name of the aggregate to be restored.

   The **-newserver** *machine* option is the DCE pathname of the File Server machine (for example, **/.../abc.com/hosts/fs2**) on which the restored aggregate is to be placed. This is necessary only if it is different from **-server**.

The **-newaggregate** *name* option is the device name or aggregate name of the aggregate on which the restored aggregate is to be placed. This is necessary only if it is different from **-aggregate**.

The **-noaction** option specifies that the command is to display the list of tapes needed to restore the aggregate without performing the actual operation.

21.6 Administering the Backup Database

A copy of the Backup Database can be installed on any server machine in a cell. The Backup Database stores two types of records that are used to track all of the backups done in the cell:

- Dump set records, which list the fileset family and the tape used in each dump set
- Administrative records, which list the fileset families, dump levels, and tape hosts

Because information about dumps is difficult to re-create, it is important that you copy the Backup Database with the **bak savedb** command periodically, perhaps weekly. When you issue the **bak savedb** command, the entire database is copied to tape. One tape needs to be designated as a Backup Database tape; when the command is issued, the tape is labeled with the name **bak_db_dump.1**.

If the Backup Database becomes damaged (for instance, if the disk that houses the database becomes damaged), you must delete the old database and restore an entirely new version from its backup tape. You can use the **bak verifydb** command to determine if the Backup Database is damaged.

Do *not* attempt to recover information from a corrupted database. Instead, use the **bos stop** command to shut down *all* **bakserver** processes. Then, remove the old Backup Database and its associated files from each machine on which it is located; the files for the Backup Database are named */dcelocal/***var/dfs/backup/bkdb.*** on each machine on which the database resides.

Once the database is removed, use **bos start** to restart *all* **bakserver** processes on the machines where they were running. Use **bak addhost** to add a tape host for the Tape Coordinator from which you will restore the

Backup Database, and use **bak restoredb** to restore the new version of the database. Re-create fileset information in the database as needed, restoring dump set information that you may have lost since the last backup of filesets; note that any fileset family, fileset family entry, or host information updated since the last backup of the Backup Database must be re-created as well.

If specific information about a dump set is accidentally deleted from the Backup Database, you can use the **bak scantape** command with the **-dbadd** option to check the backup tape used for the dump set, recover the dump set information, and add the information to the Backup Database. Do *not* use the **bak scantape** command to attempt to reconstruct the database.

21.6.1 Backing Up the Backup Database

To back up the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
3. Issue the **bak savedb** command to save the Backup Database to tape:

   $ **bak savedb** [**-tcid** *tc_number*]

4. Place the Backup Database tape in the drive, and press **<Return>** in the corresponding Tape Coordinator's monitoring window.

21.6.2 Restoring the Backup Database

To restore the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check. In addition to the usual lists, you must also be included in the **admin.bos** list on each machine on which the Backup Database is installed.

2. Verify that you have the **w** (write) and **x** (execute) permissions on the *dcelocal*/**var/dfs/backup** directory on each machine on which the Backup Database is installed.

3. Stop all **bakserver** processes with the **bos stop** command. You must stop all **bakserver** processes on all machines on which the Backup Database is installed.

4. Remove the old Backup Database by deleting the *dcelocal*/**var/dfs/backup/bkdb.*** files from each machine on which the database is installed.

5. Start all **bakserver** processes with the **bos start** command. You must start *all* **bakserver** processes that you stopped in the earlier step; you must restart the processes on the same machines on which they were previously running. When you start a **bakserver** process, an empty Backup Database is created if one does not already exist.

6. Enter the **bak addhost** command to create an entry in the Backup Database for the Tape Coordinator from which you will restore the Backup Database:

   ```
   $ bak addhost -tapehost machine [-tcid tc_number]
   ```

   The **-tapehost** *machine* option is the DCE pathname of the machine (for example, **/.../abc.com/hosts/bak1**) for which the Tape Coordinator is to be added.

7. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)

8. Issue the **bak restoredb** command to restore the Backup Database to tape:

   ```
   $ bak restoredb [-tcid tc_number]
   ```

9. Place the Backup Database tape in the drive, and press **<Return>** in the corresponding Tape Coordinator's monitoring window.

21.6.3 Recovering Specific Backup Data

Use the **bak scantape** command to extract dump set information from a backup tape and add it to the Backup Database. When you issue this command, you must place the backup tapes into the drive in sequential order. The system verifies that each tape is undamaged by checking the end-of-file markers that the Backup System inserts at the beginning and end of each fileset. If the markers are missing, the tape is assumed to be damaged and cannot be used for recovering data. To add information to the database, the entire tape must be undamaged, and the Backup Database must not contain an entry with the same dump ID as an entry being added.

To add recovered data to the Backup Database, do the following:

1. Verify that you are included in the appropriate administrative lists. If necessary, issue the **bos lsadmin** command to check.
2. If it is not already running, start the Tape Coordinator for the tape drive that you want to use with the operation. (See Section 21.2.2.1 for information on using the **butc** command to start a Tape Coordinator.)
3. Insert the first backup tape from the dump sequence into the tape drive, and issue the **bak scantape** command *without* the **-dbadd** option. Information from the tape is displayed in the Tape Coordinator's monitoring window.

   ```
   $ bak scantape [-tcid tc_number]
   ```

4. If the output indicates that the tape is undamaged, issue the **bak scantape** command again, *including* the **-dbadd** option. This adds the information from the tape to the Backup Database.

   ```
   $ bak scantape [-dbadd] [-tcid tc_number]
   ```

   The **-dbadd** option indicates that information extracted from the tape is to be added to the Backup Database; information is added only if the tape is undamaged and the Backup Database does not have an entry with the same dump ID as an entry being added.

21.7 Displaying and Canceling Operations in Interactive Mode

When you issue a command in interactive mode, the resulting operation is assigned a unique job ID number. While in interactive mode, you can use the **bak jobs** command to list job ID numbers and status information about all of the operations currently executing or pending in the queue on a tape drive (operations that do not involve tapes execute immediately and do not appear on the list). You can use the job ID number of an operation (or its dump set name if it is a dump) with the **bak kill** command to cancel an operation that is executing or that is in the queue.

If you cancel an operation that is in the queue, it is removed from the queue with no other effects. Furthermore, if you cancel a tape labeling or tape scanning operation as it executes, the operation simply terminates with no further effects. However, canceling a dump or restore operation while it executes can produce inconsistencies on a backup tape or in the file system.

If you cancel a backup operation while it is executing, all filesets written to tape *before* the **kill** signal is received are complete and usable on the tape. However, filesets being written *when* the signal is received may be incomplete and *should not be used.*

If you cancel a restore operation while it is executing, all completely restored filesets are online and usable. However, because most restore operations require data from multiple tapes (a full dump tape and one or more incremental dump tapes), most filesets are usually not completely restored. If the **kill** signal occurs before the system accesses all of the necessary tapes, most filesets are not restored to the desired date or version and *should not be used.*

If the interrupted restore operation is overwriting one or more existing filesets, the filesets can be lost entirely; however, the data being restored still exists on tape. In general, to avoid the inconsistencies that can result from an interrupted restore operation, reinitiate the restore operation.

21.7.1 Displaying Operations in Interactive Mode

Issue the **bak jobs** command to determine the job ID number of an operation. For an operation to appear in the output from the **bak jobs** command, you must have initiated the operation in interactive mode, and you must still be in interactive mode. No privileges are required to display an operation with the **bak jobs** command. (See Section 21.2.2.3 for more information about interactive mode).

```
bak> jobs
```

If no operations are pending or executing, the prompt returns immediately. Otherwise, the output reports the following information for each job. The output is very similar to that produced by the **bak status** command.

- The job ID number.
- A name describing the operation. One of the following operation names is displayed for each job:
  - `Dump` (*dump_set*) for a backup operation initiated with the **bak dump** command; *dump_set* is the name of the dump set in the form *fileset_family_name.dump_level*.
  - `Restore` for a restore operation initiated with the **bak restoreft** or **bak restoredisk** command.
  - `Labeltape` (*tape_label*) for a tape labeling operation started with the **bak labeltape** command; *tape_label* is the tape label specified with the **bak labeltape** command's options.
  - `Scantape` for a tape scanning operation initiated with the **bak scantape** or **bak readlabel** command.
  - `SaveDb` for a **bak savedb** operation.
  - `RestoreDb` for a **bak restoredb** operation.
- The number of kilobytes transferred so far (from the file system to tape for a dump operation, or from tape to the file system for a restore operation).
- For a dump operation, the string `fileset` followed by the name of the fileset currently being dumped; for a restore operation, the string `fileset` followed by the name of the fileset currently being restored.

- A message indicating the status of the operation. No message is displayed if the operation is executing normally.
  - The `[abort request]` message means the **bak kill** command was issued but the operation is not yet canceled.
  - The `[abort sent]` message means the operation is canceled; once the system removes it from the queue or stops its execution, the operation no longer appears in the listing from the **bak jobs** command.
  - The `[butc contact lost]` message means the **bak** program temporarily lost contact with the Tape Coordinator executing the operation.
  - The `[drive wait]` message means the operation is waiting for the specified tape drive to become free.
  - The `[operator wait]` message means the Tape Coordinator is waiting for the operator monitoring the command's execution to insert a tape in the drive.

21.7.2 Canceling Operations in Interactive Mode

Issue the **bak kill** command to cancel an operation. Use the **bak jobs** command to determine the job ID number of the operation to be killed. No privileges are required to cancel an operation with the **bak kill** command. The command can be issued only in interactive mode. (See Section 21.2.2.3 for more information about interactive mode.)

`bak>` **kill** {*jobID* | *dump_set*}

The *jobID* argument is the unique job ID number of the operation to be canceled; the *dump_set* argument is the name of the operation in the form *fileset_family_name.dump_level* if it is a dump. Use either argument to indicate the operation to be canceled.

Chapter 22

Monitoring File Exporters with the scout Program

The **scout** program can help you monitor the File Exporter running on any File Server machine. When you use **scout**, it periodically collects statistics from the File Server machines that you specify and displays the information in a graphical format. The program has several useful features:

- You can monitor the File Exporters on several File Server machines, including the local machine and machines in other cells, from one location.
- You can set attention thresholds for many of the statistics. When a value exceeds the specified threshold, **scout** highlights it with reverse video; when the value drops below the threshold, **scout** removes the highlighting.
- If the File Exporter on a machine does not respond to **scout**'s probes, the program highlights the machine's name on the screen and blanks out the other fields for the machine. When the File Exporter returns to service, the machine name and the statistics are again displayed normally.

22.1 An Overview of the scout Program

You can run **scout** from any machine configured as a DFS client or server. In a standard configuration, the binary file for **scout** is stored in the *dceshared*/**bin/scout** file. Both terminals and windowing systems that emulate terminals can display **scout**'s statistics; the display appears best on systems that support reverse video and cursor addressing.

When using **scout**, set the **TERM** environment variable (or its equivalent) to the correct terminal type or to one with characteristics similar to the actual ones. Do not resize the window while **scout** is running; the program does not adjust to the new dimensions if the window is made larger, and, if it is made smaller, the columns may not align properly. To resize the window, stop **scout**, resize the window, and then restart the program.

Although no special privileges are required to run **scout**, it is useful primarily for system administrators who need to monitor File Exporter usage. While **scout** imposes only a minimal burden on the File Exporter on a File Server machine, you may wish to place the binary file for the program in a secure directory that is available only to administrative users. Other users are then prevented from needlessly running the program.

You can run multiple **scout** processes from a single machine; because the **scout** program must run in the foreground, each instance runs in a separate, dedicated window. You can also run **scout** on several machines and view the output on a single machine. To do this, open several windows on one machine, log into different remote machines in each window (by using **telnet** or an appropriate program), and run **scout** in the windows. You may find it useful to include the **-host** option with the **scout** command; this option marks each window with the name of the host machine that is running the program.

In most cells, all File Server machine names share the same basename, or common DCE prefix. For example, File Server machines in the **abc.com** cell have DCE pathnames like **/.../abc.com/hosts/fs1**, **/.../abc.com/hosts/fs2**, and so on. If all of the machines in a **scout** window have the same DCE pathname prefix, you can use the **-basename** option with the **scout** command. You can then specify the unique part of each machine name (**fs1** or **fs2**) and specify the basename (**/.../abc.com/hosts**) only once. You do not have to specify the / (slash) that separates the prefix from the unique part of each machine name; it is included automatically with the **-basename** option.

22.2 The scout Screen

The output generated by the **scout** program appears in the following three main regions on the screen:

- The banner line at the top of the screen displays the word `scout`, indicating that the program is running; it can display additional information if you include the following options:
  - The **-host** option displays the name of the machine executing **scout**.
  - The **-basename** option displays the common DCE pathname prefix of the File Server machines being monitored.
- The statistics display region constitutes the majority of the window. In this region, **scout** displays the statistics gathered for each File Exporter; it displays each File Exporter on its own line. The area is divided into six columns, with the following labels and information:
  - `Conn`: The number of connections with clients. This column displays the number of RPC connections open between the File Exporter and client machines. This number should equal or exceed the number in the `Ws` column, which shows the number of active client machines. The number in the `Conn` column can exceed the `Ws` number because each user on a machine can have several separate connections open at once and because one client machine can handle several users.
  - `Fetch`: The number of bytes fetched from the File Exporter. This column displays the number of bytes of data, in 64-kilobyte units, that client machines have fetched from the File Exporter since it started. The number is reset to 0 (zero) whenever the File Exporter restarts.
  - `Store`: The number of bytes that are stored by the File Exporter. This column displays the number of bytes of data, again in 64-kilobyte units, that client machines have sent to the File Exporter for storage since it started. The number is reset to 0 (zero) whenever the File Exporter restarts.
  - `Ws`: The number of active client machines. This column contains the number of client machines (typically workstations) that have communicated with the File Exporter in the last 15 minutes. This number is usually smaller than the value for `Conn` because a single client machine can have several connections open to one server.

— *unlabeled*: The File Server machine name. This unlabeled column contains the name of the File Server machine where the File Exporter is running. Names are shortened to display only the first 10 characters of the unique part of each machine name, which are the characters that follow the **hosts** element of a DCE pathname. If two or more machines from different cells have a common name (for example, **/.../abc.com/hosts/fs1** and **/.../def.com/hosts/fs1**), the name of each cell followed by a colon is displayed before the name of each machine (**abc.com:fs1** and **def.com:fs1**).

— `Disk attn`: The disk usage. This column displays the number of available kilobyte blocks on each DFS aggregate on the File Server machine. For example, a display of `/dev/lv01:8949` indicates that the aggregate **/dev/lv01** has 8949 kilobyte blocks free. If the window is not wide enough for all of the aggregate entries, **scout** automatically creates subcolumns for the information.

The label on this column indicates the threshold value at which entries become highlighted. By default, **scout** highlights the entry for any aggregate that is over 95% full. Therefore, the default label for this column appears as `Disk attn:> 95% used`.

For all columns except the fifth (the File Server machine name), you can use the **-attention** option to set a threshold at which entries in the column are highlighted. This notifies you that a certain value is exceeded. `Disk attn` is the only statistic with a preset default.

- The message/probe line at the bottom of the screen indicates how many times **scout** probed the File Exporters for statistics. By default, **scout** probes every 60 seconds; you can use the **-frequency** option to specify a different rate of time.

22.3 Setting Attention Thresholds

The **-attention** option can be used to set the threshold value for all but the File Server machine column in the display region. Any threshold that you set applies to all of the entries in a column; you cannot set thresholds on a per-machine basis.

You can use more than one argument with the **-attention** option; each argument is a combination of a statistic and a threshold. Legal values for statistic/threshold pairs are as follows:

- The **conn** *connections* argument sets the threshold for the maximum number of connections that the File Exporter can have open to client machines before the value is highlighted. The highlighting is removed when the value goes below the threshold.
- The **fetch** *bytes_fetched* argument sets the threshold for the maximum number of bytes of data that clients can fetch from the File Exporter before the value is highlighted. Enter a threshold for this statistic in 64-kilobyte units. For example, to have **scout** highlight this value when it equals or exceeds 128 kilobytes, specify a threshold of **2**. The highlighting is removed when the File Exporter is restarted.
- The **store** *bytes_stored* argument sets the threshold for the maximum number of bytes of data that clients can send to the File Exporter for storage before the value is highlighted. Enter a threshold for this statistic in 64-kilobyte units. For example, to have **scout** highlight this value when it equals or exceeds 128 kilobytes, specify a threshold of **2**. The highlighting is removed when the File Exporter is restarted.
- The **ws** *active_client_machines* argument sets the threshold for the maximum number of active client machines that the File Exporter can serve before the value is highlighted; *active* indicates those machines that communicated with the File Exporter in the past 15 minutes. The highlighting is removed when the value goes below the threshold.
- The **disk** *percent_full%* argument sets the threshold for the maximum percentage of an aggregate that can contain data before the value is highlighted. This threshold is applied to all exported aggregates and partitions on a File Server machine being monitored. Legal thresholds are integers from 0 to 99; the default is `95% used`. You *must* enter the % (percent sign) with this threshold; if the % (percent sign) is absent, **scout** interprets the number as a number of kilobyte blocks.

  or

  The **disk** *minimum_blocks_free* argument sets the threshold that determines the minimum number of kilobyte blocks to be available on the aggregate before the value is highlighted. This threshold is applied to all exported aggregates and partitions on a File Server machine being monitored.

To change these attention settings, you must stop and restart **scout**. In addition, **scout** does not store the settings from previous instances; you must specify the desired settings each time you start the program.

You cannot set any threshold or control the highlighting in the column where File Server machine names are displayed. If the File Exporter on a machine does not respond to **scout**'s probes, the name is automatically highlighted and the values for that machine in the other columns are removed. A lack of response can indicate a File Exporter or machine crash or a network outage.

When a machine resumes responding to **scout**'s probes, its name is displayed without highlighting with the other values on its line of the display. If all of the machine names are highlighted simultaneously, a network outage has possibly disrupted the connections between the File Server machines and the client machine running **scout**.

The following examples demonstrate the different types of **-attention** settings. The first example causes **scout** to highlight entries in the `Conn` column that exceed 100, entries in the `Ws` column that exceed 20, and entries in the `Disk attn` column that reflect aggregate fullness usage of 75% or more on the machines named **/.../abc.com/hosts/fs1** and **/.../abc.com/hosts/fs2**:

```
$ scout -server /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 -attention conn
        100 ws 20 disk 75%
```

The second example is identical to the previous example, except that **scout** highlights entries in the `Disk attn` column that fall below 5000 free blocks:

```
$ scout -server /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 -attention disk
        5000 ws 20 conn 100
```

The third example causes **scout** to highlight entries in the `Fetch` column that exceed 1 megabyte (1024 kilobytes):

```
$ scout -server /.../abc.com/hosts/fs1 /.../abc.com/hosts/fs2 -attention fetch 16
```

22.4 Using the scout Program

Start the **scout** program by issuing the **scout** command to initialize it in each window in which you want it to run. No further commands can be issued in the window as long as the program is running. To stop **scout**, enter the interrupt command (**<Ctrl-c>** or its equivalent) for your system in the window in which **scout** is running.

22.4.1 Starting the scout Program

To start the **scout** program, do the following:

1. Open a command shell window for each instance of **scout** you want to run.
2. Initialize **scout** in each window; note that you will not be able to issue any further commands in the window as long as **scout** is running:

   $ **scout -server** *machine...* [**-basename** *common_prefix*] [**-host**] [**-frequency** *seconds*] [**-attention** *stat/threshold_pair...*] [**-debug** *filename*]

   The **-server** *machine* option names each File Server machine whose File Exporter is to be monitored. Provide the DCE pathname of each machine (for example, **/.../abc.com/hosts/fs1**), unless the **-basename** option is used; if the **-basename** option is used, provide only the unique suffix of each machine name, omitting the common DCE pathname prefix.

   The **-basename** *common_prefix* option specifies the DCE pathname prefix common to the File Server machines specified with **-server**. If the basename is specified, it is displayed in the banner line.

   The **-host** option displays the name of the machine running **scout** in the banner line; this is useful if you are logged into the machine remotely.

   The **-frequency** *seconds* option indicates how often **scout** is to probe the File Exporters. Specify a positive integer as a value in seconds; the default is 60 seconds.

The **-attention** *stat/threshold_pair* option specifies a list of attention settings (statistic and threshold pairs); **scout** highlights any value for a statistic that exceeds its threshold. (See Section 22.3 for a discussion of legal values for this argument.)

The **-debug** *filename* option enables debugging output and directs it to the specified *filename*. Provide a complete pathname for *filename*; the current working directory is used by default. If this option is omitted, no debugging output is written.

The following example causes **scout** to monitor the File Exporters on the File Server machines named **fs1** and **fs2** in the cell named **abc.com**; the **-basename** option is used, so the common DCE prefix is specified only once, and it appears in the banner line. The **scout** program probes the File Exporters every 30 seconds and prints debugging information to the file named **/.../abc.com/fs/usr/terry/scout.one**.

```
$ scout -server fs1 fs2 -basename /.../abc.com/hosts -frequency 30
    -debug /.../abc.com/fs/usr/terry/scout.one
```

The following example again instructs **scout** to monitor the File Exporters on the **fs1** and **fs2** machines in the **abc.com** cell; the **-basename** option is again used to indicate the common prefix. The **-host** option is used, so the name of the machine running **scout** appears in the banner line with the basename prefix. The **scout** program highlights an entry in the `Fetch` column if more than 1 megabyte of data is fetched from a File Exporter; **scout** highlights an entry in the `Store` column if more than 512 kilobytes of data are stored by a File Exporter.

```
$ scout -server fs1 fs2 -b /.../abc.com/hosts -host
    -attention fetch 16 store 8
```

22.4.2 Stopping the scout Program

To stop the **scout** program, enter the interrupt command (**<Ctrl-c>** or its equivalent) for your operating system in the **scout** window.

Part 3

DCE Diskless Support Service

Chapter 23

Introduction to DCE Diskless Support Service

The DCE Diskless Support Service helps you to connect client hosts that do not have local disk drives or other storage to a DCE cell. On regular machines, local storage is used for operating system files, the root file system, memory swapping areas, and the DCE Distributed File Service (DFS) cache storage. On a diskless machine, this storage, except for cache, is located on remote server hosts. These remote servers can be on the same Local Area Network (LAN) as the client or on different LANs that are connected through message-passing machines that are called *gateways*. DFS cache storage can be configured in a section of the client's memory.

The Diskless Support Service provides the tools that allow the diskless client host to do the following:

- Acquire an operating system
- Obtain configuration information
- Connect to the DCE Distributed File Service (DFS)
- Perform remote swapping

Each of these categories of diskless support is described briefly here and in more detail in the following sections. The process of planning, installing, and configuring the server hosts is described in the following chapters.

With the Diskless Support Service, a diskless client host locates a copy of its operating system on a remote boot server host, loads it into memory, and starts it executing.

Once booted, the diskless client obtains its necessary configuration data from a DCE Diskless Configuration (DLC) server. This data includes the location of the client's root file system and the configuration of the file system cache storage in the client's memory. If needed, it includes the location of the client's remote swap server. It can also include other information that is needed by the client's operating system.

A diskless client needs a storage location for user and system files. DCE provides the mechanism for the client to have a file system on a remote file server host.

Similarly, a diskless client needs a location for swapping storage, which is a place for the operating system to store currently idle blocks of memory that are used by one process so the memory can be used by another process. In DCE, this storage space can be provided in the client's root file system or by a remote swap server host.

The boot, file, and swap servers can be on the same physical host, or they can be divided among two or three machines.

23.1 The Diskless Boot Process

The booting process for a diskless client typically proceeds as follows:

1. A diskless client executes a program from read-only memory (BOOTROM) when it is powered on. This program sends a boot protocol (BOOTP) broadcast message on all attached LANs, seeking a boot server.
2. A boot server responds with the name of the client's boot file on the server, the Internet Protocol (IP) addresses of the client and the boot server, and the client's hostname. Optionally, the boot server may return the network netmask and the IP address of the network gateway.
3. The client sends a Trivial File Transfer Protocol (TFTP) request to the boot server, asking for the boot file.

4. The server transmits the file to the client, which loads the file into memory.
5. The client executes the file as its operating system.

Three programs are provided to perform these tasks:

- The **client** program is the user-space version of the client's BOOTROM program. It acts as a BOOTP and TFTP client. In order to run on a diskless client, this program must be modified as described in the *OSF DCE Porting and Testing Guide*, and must be installed in the client host's BOOTROM. This installation is provided by the vendor who supplies the client machine or adapts its software for your DCE environment.
- The **bootpd** program acts as a BOOTP server. It executes on the boot server host, and on the gateway from the client host's LAN, if necessary. It responds to a broadcast request for a boot file server and returns the client's boot filename, the IP addresses of the client and boot server, and the client's hostname. Optionally, the boot server may return the network netmask and the IP address of the network gateway.
- The **tftpd** program acts as a TFTP server. It executes on the boot server host. It transfers a copy of the client's boot file to the client.

(See the **bootpd(8dskl)**, **client(8dskl)**, and **tftpd(8dskl)** reference pages in the *OSF DCE Administration Reference*. See Chapter 24 for a description of setting up and operating diskless booting.)

23.2 Diskless Configuration

The configuration process for a diskless client consists of a call from the client's operating system to a DLC server, using the boot server address and the hostname that were received in the boot reply packet from the boot server. The DLC server returns the value of the hostname, which includes the location of the client's root file system and the configuration of the file system cache storage in the client's memory. If needed, it includes the location of the client's remote swap server. It can also include other information that is needed by the client's operating system.

The client operating system calls DFS with the file system and cache information. DFS establishes the client's root file system and sets up the DFS cache storage in the client's memory.

(See the **dlcd(8dskl)** and **dlctab(8dskl)** reference pages in the *OSF DCE Administration Reference*. See Chapter 25 for a description of preparing configuration data and storing it in the DLC server database.)

23.3 Diskless Swapping

Swapping is the process of copying sections of memory to mass storage so that the memory space can be used by a process for other data. Later, the stored data can be returned to memory for further use. The term ''swapping'' is used here to mean any sort of temporary transfer between memory and backing storage. It includes and does not distinguish between the swapping of entire processes and the paging of memory blocks.

UNIX operating systems usually swap in one of two ways: to files or to devices. The two are administered differently, both on regular hosts with local storage and on diskless client hosts with remote storage.

Swapping to files
: On hosts with local storage, this method is usually handled by local file interfaces. For diskless clients, DCE provides remote versions of these operations, so the swap files can be located on the diskless client's remote file system. No remote swap server is needed.

Swapping to devices
: On hosts with local storage, this method is usually handled by a device driver in the system's block or character device switch tables. For diskless clients, DCE provides a replacement driver that sends swap requests across the network to a swap server.

The routines and programs that are provided with the DCE Diskless Support Service are intended for diskless clients that swap to devices. There are three components:

- The swap driver is intended for the diskless client's operating system. It has the same interface as a device driver, but, instead of sending its swap requests directly to a device, it sends them to a remote swap server. This

driver is installed in the client operating system by the vendor who supplies it or adapts it for your DCE environment.

The swap driver obtains its swap server location from the DLC server subentry data that was previously acquired in the configuration phase.

- The **dswd** swap server daemon executes on the swap server host and handles requests from diskless clients. The **dswd** daemon allows each client to view its swap space as a single block of addresses, starting at 0 (zero), even though the daemon is actually mapping the requests to a number of devices or ordinary files on the swap server. There is no requirement that swap space be contiguous for any client.

  Note: The **dswd** daemon can use both device files and ordinary files in the swap space that it manages. This is completely independent of the fact that the client operating system thinks it is swapping only to a device.

- The **dsw_adm** command configures the swap server. It can execute on any host in the cell, including a diskless client. The **dsw_adm** command tells **dswd** which device and ordinary files it can use for swapping and which clients it can accept requests from. Until **dsw_adm** commands have configured the swap server, **dswd** does not respond to any client requests.

The swap requests that are issued by a diskless client fall into two categories. Input and output requests move memory pages to or from swap storage. Allocation requests ask the swap server for initialization, for more or less storage, and for termination.

Before a diskless client can read or write any swap space, it must allocate that space on the swap server. A client may allocate space in one of two ways: dynamically or statically. A dynamic client is one that can request more swap space after it has started swapping. Some dynamic clients can return space when they are done with it. A static client does not have this capability. It needs to have all of its swap space allocated at once when it initializes.

When a diskless client begins operation, it sends an initialization request to the swap server. This request indicates whether the client is dynamic or static. A static client is given all of its space immediately. A dynamic client is given an initial allocation, and requests more (or less) space as it requires it, up to a set maximum. When a client shuts down, it can send a termination request to the swap server, releasing all of its swap space. If a client fails to release its swap space and then sends another initialization

request, the swap server releases all of the old allocated swap space and allocates new space for the client. This effectively handles the problem that could occur if a client crashed without releasing its swap space.

Once a client is granted a space allocation, the physical space in the server's device and ordinary files is reserved for the client, regardless of the actual swap requests. This prevents the server from using allocated but unused space from one client to satisfy requests from another client. For example, if a swap server has 20 megabytes of swap space, it cannot accept two allocation requests of 15 megabytes each, even though the two clients do not actually use any space at all. The first request is honored; the second is rejected.

(See the **dsw_adm(8dskl)** and **dswd(8dskl)** reference pages in the *OSF DCE Administration Reference*. See Chapter 26 for a description of configuring the swap server.)

23.4 Device Files

Device-special files that are defined on the remote file system that is used by the diskless client host (for example, in the **/dev** directory) refer to devices (such as terminals) on the client host, not on the server host that provides the file system.

23.5 Machine-Specific Files

If two or more clients (diskless or not) share all or part of a file system, the **@host** and **@sys** operating system macros can be used to identify client-host-specific or operating-system-specific files. (See Part 2 of this guide for details.)

23.6 Documentation References

References to a DCE reference page for a command, file, or function always include a section number that contains an abbreviation for the module it is part of. For example, **bootpd(8dskl)** is the designation for the documentation of the **bootpd** daemon in section 8 of the DCE Diskless Support Service reference pages.

Sometimes, the procedures that are described in this module use commands and files that are not part of the DCE offering but are assumed to exist as part of the host system. Generally, section references that do not include a DCE module abbreviation indicate documentation that you will find in your own system reference pages. The section numbers shown refer to OSF/1 reference manuals. Other systems can use different numbers.

Chapter 24

Managing Diskless Booting

For the system administrator, managing the boot component of the DCE Diskless Support Service consists chiefly of setting up the **bootpd** and **tftpd** daemons on the appropriate boot server and gateway hosts in the DCE cell, creating the **bootptab** boot configuration file for each server, and adding the appropriate entries to the standard **inetd.conf**, **passwd**, and **services** files. Note that the installation of the **client** program in the BOOTROM of the diskless client is handled by the vendor who supplies the machine or adapts it for your DCE environment.

For a diskless client host, the boot component of the DCE Diskless Support Service consists of the following programs and files:

- The **client** program in the BOOTROM of the client host
- The **bootpd** and **tftpd** programs and the **bootptab** file on a boot server host
- The **bootpd** program and the **bootptab** file on one or more gateway hosts, if needed

Typically, the clients and boot servers are on the same local network, eliminating the need for a gateway to other networks.

24.1 Choosing the Boot Server Hosts

A boot server host needs to be one of the nodes in the DCE cell that the diskless client hosts belong to. At minimum, it must include the **bootpd** and **tftpd** daemons that serve the diskless clients. It also requires access to some form of secondary storage, such as a disk drive.

Beyond these requirements, the choice of one or more boot server hosts and their relationship to the file server and swap server hosts is largely arbitrary, depending on the factors of administrative convenience and optimal throughput. The following points can be helpful in making and revising your choices:

- In a simply organized system, the boot server, file server, and swap server for a client can all reside on the same machine. This provides ease of identification for the user and the administrator because all of the basic data resides on the same hardware.
- On a busy network, you may be able to improve efficiency by putting the boot server, file server, and swap server on separate hosts.
- Boot servers can be spread over several hosts to reduce the impact of many simultaneous power-ups and to make it possible for some clients to run if one boot server is unavailable.
- If there are several boot servers on a local network, you can set them up so that each one serves a separate group of clients; that is, you put a host entry for a client in only one **bootptab** boot configuration file. That way, a client gets a boot reply from only one server.
- Alternatively, you can provide redundancy and therefore availability by duplicating the client data on several boot servers. This would be most functional if the file and swap servers were on different machines from the boot server.
- Because regular systems provide all of the operating system, local file system, and swapping facilities on the same host, the effect of diskless clients on file and swap servers is related most to the number of active users that the clients support at one time. Network and server activity that is required for local file and swap access can require more computing resources, while the fact that the principal computing activity is occurring away from the server host can reduce the load.

- Use caution when configuring **bootpd** on gateways. Do not use the forwarding parameters, **-f** and **-g**, unless clients on the local network must boot from servers on foreign networks. Consider particularly whether you need the **-g** parameter, which forwards unknown boot requests to foreign networks. If the gateway is not a boot server or is not the only boot server on the local network, it can end up forwarding requests that are actually handled on the local network, resulting in unnecessary network traffic. Note that the **bootpd** program can be run with the **-f** or **-g** parameters only on a gateway.

24.2 The Diskless Client Host

For normal operations, a diskless client host requires the supplied user-space **client** program to be converted to power-up BOOTROM code. This conversion is handled by the vendor who supplies the client machine or adapts it for your DCE environment. However, you may be able to use the user-space client program to test the client/server interaction up to the reception of the boot file on the client host. You can install the user-space **client** program on any convenient host in the DCE cell. (See the *OSF DCE Porting and Testing Guide* for details.)

24.3 Setting Up a Boot Server

The installation of the boot component on a boot server involves installing the **bootpd** and **tftpd** programs, creating the **bootptab** boot configuration file, creating the pseudo-user **tftp**, installing the boot files for the clients, and adding entries to the **services** and **inetd.conf** files. The following procedure gives you step-by-step instructions on how to install the boot server.

Note: A boot server host can also be a gateway. If it is a gateway, combine this procedure with the one in Section 24.4.

1. Make sure that the **bootpd** and **tftpd** programs are located in the **/etc** directory in your local file system.

Note: The **/etc** directory is the customary location for these files. However, you can install them anywhere in the local file system of the boot server if you modify the subsequent steps in this procedure and the next procedure in Section 24.4 to match each other.

2. Add the following lines to the **/etc/services** file to define the **bootps**, **bootpc**, and **tftpd** service ports. (See the **services(4)** reference page in your system documentation for further information.)

```
bootps          67/udp          # bootp server port
bootpc          68/udp          # bootp client port
tftp            69/udp          # tftp client port
```

3. The **bootpd** and **tftpd** programs are normally run as daemons under the control of the **inetd** daemon. To do this, add the following lines to the **/etc/inetd.conf** configuration file. (See the **inetd(8)** and **inetd.conf(4)** reference pages in your system documentation for further information.)

```
bootps dgram udp wait root /etc/bootpd bootpd [bootpd-parameters]
tftp   dgram udp wait root /etc/tftpd  tftpd [tftpd-parameters]
```

Each line can be followed by optional parameters, which are described in the **bootpd(8dskl)** and **tftpd(8dskl)** reference pages in the *OSF DCE Administration Reference*. For **bootpd**, the defaults are an idle timeout of 15 minutes, the boot configuration file **/etc/bootptab**, the dump file **/etc/bootp.dump**, and a diagnostic output level 0 (zero). For **tftpd**, the defaults are 5 seconds for packet retransmission and 25 seconds for packet transmission failure.

4. Using your normal procedures, create the pseudo-user **tftp** in the local file system. The home directory of this user is the base location for the client boot files. The **/etc/passwd** file entry for **tftp** should look like the following:

```
tftp:*:510:20:tftp server:/users/tftpdir:/bin/false
```

The fields in the **/etc/passwd** file entry for **tftp** are described in Table 24-1.

Table 24–1. Password File Entry for Pseudo-User tftp

Field	Value	Notes
Login name	**tftp**	Required
Password	*	Asterisk to prevent direct logins
User ID	*anything*	Must be unique
Group ID	*guestid*	Group ID number for **guest**
Description	*anything*	—
Home directory	*anything*	Full pathname required
Login shell	**/bin/false**	—

5. Install the boot files in the **tftp** home directory or subdirectories. Set their file permissions with **chmod** so they are readable by **others**. The command **chmod 004** *filename* sets the read bit for **others** and clears all the other bits.

   These files are the binary code that is executed by diskless clients as their operating systems. They are provided by the vendor who provides the client machines or adapts them for your DCE environment. You have a number of options. If several of your diskless clients are on the same hardware and operating system, they each may be able to use the same boot file, reducing the need for multiple copies of the same file. For those clients that need individual boot files, you can use the same generic name, thus simplifying the **bootptab** boot configuration file entries, which you can distinguish for each client by adding *.hostname* as a suffix or extension.

6. Create the **bootptab** boot configuration file. (See the **bootptab(8dskl)** reference page in the *OSF DCE Administration Reference* for details.) The **bootpd** daemon checks this file to update its list of client hosts before processing each boot request packet.

   Note: It is recommended that you use the default **/etc/bootptab** as your normal boot configuration file and reserve the other names for special operations or testing.

   Each entry in the file must contain the client's hostname and Internet Protocol (IP) address. The file should contain the client's hardware type and hardware address, since it is likely that the client will specify its hardware address and not its IP address. It should also contain the home directory (actually, a subdirectory of the **tftp** home

directory), expressed as an absolute pathname, and the filename of the client's boot file. Note that the home directory and the filename of the client's boot file are not necessary if the client supplies the full pathname in its request packet.

For example,

```
myclient:ht=ethernet:ha=01.27.13.77.A6.12:ip=123.5.78.4\
        :hd=/boot:bf=kernel
```

defines a client host named **myclient** whose Ethernet hardware address is **01271377A612** (hexadecimal), whose IP address is **123.5.78.4** (dotted-decimal octets), and whose boot filename, which is rooted in the **tftp** home directory, is **/boot/kernel**.

Note: If this example and the one for the **passwd** file in step 4 are combined, the full pathname of the boot file in the file system would be **/usr/tftpdir/boot/kernel**. However, because **tftpd** performs a **chroot** (change root directory) command that makes the **tftp** home directory its effective root directory, the full pathname that is given to **tftpd** must be **/boot/kernel** for the file to be found.

24.4 Setting Up a Gateway

You need to set up **bootpd** on a gateway only if a diskless client on the local network needs to get its boot file from a boot server on a foreign network. This could happen if all of the hosts on the local network are diskless, including the gateway. If all of the diskless clients on the local network can get their boot files from servers on the local network, do not set up **bootpd** on a gateway; it is not needed, and omitting it will reduce network traffic.

The installation of the boot component on a gateway involves installing the **bootpd** program, adding entries to the **services** and **inetd.conf** files, and

creating the **bootptab** boot configuration file. The following procedure gives you step-by-step instructions on how to install a boot gateway:

Note: A gateway can also be a boot server host. If it is a boot server host, combine this procedure with the one in Section 24.3.

1. Make sure that the **bootpd** program is located in the **/etc** directory in your local file system.

   Note: The **/etc** directory is the customary location for this file. However, you can install it anywhere in the local file system of the gateway if you modify the subsequent steps in this procedure and the previous procedure in Section 24.3 to match each other.

2. Add the following lines to the **/etc/services** file to define the **bootps** and **bootpc** service ports. (See the **services(4)** reference page in your system documentation for further information.)

```
bootps          67/udp          # bootp server port
bootpc          68/udp          # bootp client port
```

3. The **bootpd** program is normally run as a daemon under the control of the **inetd** daemon. To do this, add the following entry to the **/etc/inetd.conf** configuration file. (See the **inetd(8)** and **inetd.conf(4)** reference pages in your system documentation for further information.)

   `bootps dgram udp wait root /etc/bootpd bootpd [-f] [-g]` *[bootpd-parameters]*

   Either or both of the **-f** and **-g** parameters must be specified. They are only permitted on gateways. The **-f** parameter allows request packets that specify a server hostname to be forwarded to that server. The hostname must be defined in the **/etc/hosts** hosts file, which is described in the **hosts(4)** reference page in your system documentation. The **-g** parameter allows request packets that do not specify a server hostname to be forwarded to a list of servers that are defined in the **bootptab** boot configuration file. If the gateway is also a boot server, then the packets are forwarded only if they cannot be handled locally.

The line can be followed by other options, which are described in the **bootpd(8dskl)** reference page in the *OSF DCE Administration Reference*. The defaults are an idle timeout of 15 minutes, the boot configuration file **/etc/bootptab**, the dump file **/etc/bootp.dump**, and a diagnostic output level of 0 (zero).

4. Create the **bootptab** boot configuration file. (See the **bootptab(8dskl)** reference page in the *OSF DCE Administration Reference* for details.) The **bootpd** daemon checks this file to update its list of client hosts before processing each boot request packet.

   Note: It is recommended that you use the default **/etc/bootptab** as your normal boot configuration file and reserve the other names for special operations or testing.

   If **bootpd** is run with only the **-f** parameter, the **bootptab** boot configuration file must exist, but it can be empty if the gateway is not also a boot server host.

   If **bootpd** is run with the **-g** parameter, only one entry is necessary in the **bootptab** file for the hostname **bootp.servers**. This entry must contain the hostname and the **gw** (gateway) tag fields. The **gw** tag field contains the IP addresses of the boot server hosts to whom boot request packets are forwarded for processing.

   For example,

   ```
   bootp.servers:gw=1.3.5.7  2.4.6.8  11.22.33.44
   ```

   defines the host entry **bootp.servers**, which allows the gateway to forward boot request packets to the IP addresses **1.3.5.7**, **2.4.6.8**, and **11.22.33.44**.

24.5 Authentication

Diskless booting does not contain any authentication or security facilities other than the minimal requirements that the client's hardware or IP address be known to the boot server, and that the boot file be within the **tftp** home directory tree and be readable by anyone. In order for it to be secure, it would require hardware support, which is outside the realm of DCE. This support can be added by system vendors, however, and can be integrated with the DCE Security Service. Hardware support for secure booting would involve a way of knowing the following:

- The name of the diskless machine
- The name of each boot server
- The diskless machine's Kerberos private key

Chapter 25

Managing a Diskless Configuration

The configuration of operating systems on diskless client hosts is managed through an entry in the Diskless Configuration Server (DLC) database. When a diskless client executes its boot procedure, it provides the location of the boot server and the client's hostname. Using the hostname as a key, the client host requests its configuration data from the DLC server. The DLC server must be the same machine as the boot server.

The configuration data typically contains the locations of the DCE Distributed File Service (DFS) servers, the name of the client's root file system, the parameters for the DFS cache manager, and, if the client uses one, the location of the remote swap server. The data can also include any number of other items that are useful to the particular operating system.

The diskless client uses this data to establish its initial communications with DFS, to set up its file cache in memory, to mount its root file system, and to communicate with a remote swap server, if needed.

Generally, the configuration parameters are provided by the vendor who supplies the client machine or adapts its software for your DCE environment. However, you may need to modify some parameters, such as the locations of the DFS servers, the root file system, and the remote swap server.

The root file system for a diskless client is prepared as a DFS fileset on a host system in the same cell as the client. The process is described in Part 2 of this guide. The name of the fileset is assigned to the **root_fileset** variable in the **dlctab** configuration file. The client operating system must contain a routine that establishes the fileset as the client's root file system.

The Internet Protocol (IP) addresses of the DFS server hosts and the swap server host for the client are assigned to the **dfs_servers** and **swapserver** variables, respectively, in the **dlctab** configuration file.

Several variables refer to the DFS Cache Manager: **ndaemons**, **cachestatentries**, **cachefiles**, **cacheblocks**, **cachevolumes**, **cachedcaches**, **cachesettime**, and **cachecsize**. (See Section 25.1.1 for the meaning of these variables.)

The system administrator maintains the definitions and values of the diskless configuration data in the **dlctab** configuration file. The **dlcd** daemon reads the file and creates the diskless entry and the data subentries to later be returned to a diskless client during initialization.

25.1 Preparing the Configuration File

You can prepare and maintain the **dlctab** diskless configuration file with any line-oriented text editor.

The file has two parts: type definitions and data assignments. Each part consists of named block statements that can be combined to produce a wide variety of data specifications, which are suitable for any application.

You can also use **define** statements to specify substitute data type names and **include** statements to insert the contents of other files.

The configuration file needs to have one data block statement for every diskless client host. A lookup is done via the client's hostname. In addition, common values can be entered in separate data block statements, which can be connected to the principle host entries with continuation fields.

The file also needs at least one type block statement that defines the variables that are assigned values in the data block statements. A type block statement can be continued to another type block statement in the same tree structure format that is described for the data block statements.

Each data block statement contains one reference to a type block statement. The data block can assign values only to variables that are defined in that type block and its continuations. Each data block continuation specifies its own type block statement.

Each data block statement with its continuations becomes a separate subentry in the diskless entry in the DLC database. The subentry key is the name of the data block statement. In the example in Section 25.1.1, two data blocks are defined. One block continues to the other. Consequently, there are two subentries in the DLC database: one for the uncontinued block and one for the continued block that contains all of the values assigned in both blocks.

25.1.1 Configuration File Example

The following example shows the principle components of the configuration file and how they can be combined to produce a subentry for the DLC database:

```
#  The type block section starts the configuration file
define ip_addr as 4 bytes

#  Standard type block
dce1_0:cell_name=1.string: \                    # DCE cell name
        :dfs_servers=2.array 8 of ip_addr: \    # DFS file server IP addresses
        :root_fileset=3.string: \               # client's root file system
        :ndaemons=4.int32u: \                   # DFS cache manager parameters
        :cachestatentries=5.int32u: \           # "
        :ntknprocs=6.int32u: \                  # "
        :cacheblocks=7.int32u: \                # "
        :cachevolumes=8.int32u: \               # "
        :cachedcaches=9.int32u: \               # "
        :cachesettime=10.int32u: \              # "
        :cachecsize=11.int32u: \                # "
        :swapserver=12.ip_addr:                  # swap server IP address
        :principle=13.array 8 of string: \      # principle name for dfs_servers
        :uuid=14.string: \                      # UUID for dfs_servers
```

```
begin data

#  The data block section follows "begin data"
ourcell:dce1_0:cell_name="soft_cell": \
        :dfs_servers=1.2.232.54,1.2.232.17: \
        :ndaemons=2:cachestatentries=300:ntknprocs=2: \
        :cacheblocks=50:cachevolumes=50:cachedcaches=100: \
        :cachesettime=0:cachecsize=8192:

myclient:dce1_0:root_fileset="FS_myroot" \
        :swapserver=1.2.232.83:cont@=ourcell
```

The first line in the example is a comment, which begins with a # (number sign) character. Comment lines and blank lines can occur between statements and are ignored.

The first statement is a **define** statement, which defines the name **ip_addr** to represent the data type **4 bytes**. Wherever **ip_addr** is found as a variable type in subsequent type block statements, it is replaced by **4 bytes**.

The second statement is a type block, which has been continued over several lines of input. You can see that the continuations are symbolized by the \ (backslash) character. Each line, continued or not, can contain trailing spaces and an optional comment, which begins with the # (number sign) character.

The fields in the type block statement are separated by **:** (colon) characters. Empty or blank fields, such as the ones between the colon at the end of one line and the colon at the beginning of the next, are ignored. A field can be up to 2048 characters long.

The first field is the name of the type block, in this case, **dce1_0**. The next fields each define a triplet of variable name, variable number, and variable type. The name is used to refer to the variable in the data block section. The number is used by the client operating system to refer to the variable in the subentry data in the DLC database. Numbers 0 through 99 are reserved. The type determines what can be assigned to the variable in the data block.

The triplets have the relationships listed in Table 25-1:

Table 25–1. Configuration File Variables

Variable Name	Variable Number	Variable Type	Type Description
cell_name	1	**string**	Quoted string
dfs_servers	2	**array 8 of 4 bytes**	8-element array of 4 dot-separated bytes
root_fileset	3	**string**	Quoted string
ndaemons	4	**int32u**	Unsigned 32-bit integer
cachestatentries	5	**int32u**	Unsigned 32-bit integer
ntknprocs	6	**int32u**	Unsigned 32-bit integer
cacheblocks	7	**int32u**	Unsigned 32-bit integer
cachevolumes	8	**int32u**	Unsigned 32-bit integer
cachedcaches	9	**int32u**	Unsigned 32-bit integer
cachesettime	10	**int32u**	Unsigned 32-bit integer
cachecsize	11	**int32u**	Unsigned 32-bit integer
swapserver	12	**4 bytes**	4 bytes
principle	13	**array 8 of string**	8-element array of strings
uuid	14	**string**	Character array

The third statement, **begin data**, marks the start of the data block section.

The fourth statement is a data block, which is in a similar form to the type block statement that is above it. The first field is the name of the data block. This name is also used as the name or key of the subentry that is formed from this data block statement. The second field refers to a type block. The remaining fields assign values to variables. Only the variables that are defined in the referenced type block are eligible in the current data block.

The fifth statement is also a data block. In addition to the features of the previous data block, this one ends with a continuation field that connects it to the data block that is named in the field. In this case, the second data block includes the assignments from the first data block, and the values from both blocks are stored in the second block's subentry in the DLC database.

Note that while all the variables are defined here in one type block, they can be assigned values in separate data blocks, allowing global values to be assigned in one data block, and host-specific values to be assigned in another.

Note that *all* data block names *must* be hostnames. A client is only allowed to access information for that client, thus all information for that client is keyed off of the hostname. If multiple clients use the same configuration data, you can use continuation fields to link all of the entries to one data block. Using the previous example, you would create one entry for **ourcell** that contains all of the configuration information. You would then create an entry for each client (using the hostname for the data block name) that contains a continuation field to **ourcell**.

This configuration data results in two subentries in the DLC database: one subentry named **ourcell** that contains the values assigned in the first data block, and one named **myclient** that contains the values that are assigned in both data blocks. By specifying multiple data blocks in the form of **myclient**, you can make subentries for a number of diskless clients that can have different root file and swap server locations.

25.1.2 Syntax Summary

The following subsections provide a summary of the syntax of the statements in the configuration file. The complete description and the rules are given in the **dlctab(8dskl)** reference page in the *OSF DCE Administration Reference*.

25.1.2.1 The Type Block and Data Block Statements

A type block statement has the following format:

type-block-name [**:** *var-name* = *var-num* **.** *var-type*]...
  [**: cont@** = *type-block-name*]

A data block statement has the following format:

data-block-name **:** *type-block-name* [**:** *var-name* = *var-data*]...
  [**: cont@** = *data-block-name*]

where:

data-block-name is the name of the data block.

type-block-name is the name of the type block.

var-data is a data value that is assigned to the variable.

var-name is an arbitrary variable name that is used only in the configuration file.

var-num is a number (1 to 255) that is used by the client operating system to identify the configuration value.

var-type is the data type of the variable.

The *data-block-name*, *type-block-name*, and *var-name* variables all start with a letter, underscore, or dollar sign (**A** to **Z**, **a** to **z**, **_**, and **$**) and continue with a letter, underscore, dollar sign, or decimal digit (**0** to **9**).

The *var-type* variable can be one of the following *simple-type* values:

int8 A signed 8-bit integer

int8u An unsigned 8-bit integer

int16 A signed 16-bit integer

int16u An unsigned 16-bit integer

int32 A signed 32-bit integer

int32u An unsigned 32-bit integer

length **bytes** A sequence of dot-separated **int8u** integers

string A quoted string of 0 to 65,535 characters

or an array of *simple-type* values:

array *length* **of** *simple-type*
A fixed length array of *simple-type* elements

array of *simple-type*
A variable length array of 0 to 65,535 *simple-type* elements

The *length* variable has the range 1 to 65,535. However, fields that are between colons are limited to 2048 characters, so the practical value of *length* is also limited.

The *var-data* variable for a simple type can have one of the following *simple-value* formats:

integer A decimal, octal, or hexadecimal integer that is usable for the **int*** type.

'character' A character in apostrophes whose numeric range is 0 to 255 that is usable for any **int*** type.

"[string]" A quoted string that is usable for the **string** simple type.

integer[**.***integer*]...
A dot-separated sequence of **int8u** integers that is usable for the **bytes** simple type. Omitted values are set to 0 (zero).

or for an array type:

simple-value [**,** *simple-value*]...
A comma-separated sequence of *simple-value* elements. Omitted values are set to 0 (zero).

The specific forms for *integer*, *character*, and *string* are described in detail in the **dlctab(8dskl)** reference page in the *OSF DCE Administration Reference*.

25.1.2.2 The include Statement

The **include** statement has the following format:

include *filename***;**

where *filename* is any filename. If the filename is not absolute, it is relative to the directory where **dlctab** resides. The file is inserted in place of the **include** statement.

25.1.2.3 The define Statement

The **define** statement has the following format:

define *substitute-type* **as** *actual-type*

where *substitute-type* has the same format as a *data-block-name*, which is described previously, and *actual-type* is any string of characters, including embedded spaces. When *substitute-type* is found in the variable type location in subsequent type block statements, it is replaced with *actual-type*.

25.2 Setting Up a DLC Server

The installation of the DLC component on a DLC server involves installing the **dlcd** program, creating the **dlctab** DLC configuration file, and adding entries to the **services** and **inetd.conf** files. The following procedure gives you step-by-step instructions on how to install a DLC server:

1. Make sure that the **dlcd** program is located in the **/etc** directory in your local file system.

   Note: The **/etc** directory is the customary location for these files. However, you can install them anywhere in the local file system of the DLC server if you modify the subsequent steps in this procedure to match this new directory.

2. Add the following lines to the **/etc/services** file to define the DLCS and DLCC service ports. (See the **services(4)** reference page in your system documentation for further information.)

   ```
   dlcs            347/udp         # DLC server port
   dlcc            348/udp         # DLC client port
   ```

3. The **dlcd** program is normally run as a daemon under the control of the **inetd** daemon. To do this, add the following line to the **/etc/inetd.conf** configuration file. (See the **inetd(8)** and **inetd.conf(4)** reference pages in your system documentation for further information.)

   ```
   dlcs dgram udp wait root /etc/dlcd dlcd [dlcd-parameters]
   ```

   Each line can be followed by optional parameters, which are described in the **dlcd(8dskl)** reference page in the *OSF DCE Administration Reference*. For **dlcd**, the defaults are an idle timeout of 15 minutes, the DLC configuration file **/etc/dlctab**, the dump file **/etc/dlcd.dump**, and a diagnostic output of level 0 (zero).

4. Create the **dlctab** DLC configuration file. (See the **dlctab(8dskl)** reference page in the *OSF DCE Administration Reference* for details.) The **dlcd** daemon checks this file to update its list of client hosts before processing each DLC request packet.

   Note: It is recommended that you use the default **/etc/dlctab** as your normal DLC configuration file and reserve other names for special operations or testing.

Chapter 26

Managing the Diskless Swap Server

Any host machine with local storage can function as a swap server for one or more diskless clients. The server must be in the same DCE cell as the clients. It runs the **dswd** swap server daemon, which is configured with **dsw_adm** commands. The commands are described in detail in the **dsw_adm(8dskl)** and **dswd(8dskl)** reference pages in the *OSF DCE Administration Reference*.

You need to create a script of commands on each swap server host that automatically starts the **dswd** daemon and configures it with **dsw_adm** commands. This script is executed as part of the swap server host's normal boot procedure, so you can include it in or call it from the host's initialization scripts.

Note: Since these commands use DCE Remote Procedure Calls (RPCs) for communications, they must be executed after the **rpcd** RPC daemon has been started. (See the **rpcd(8rpc)** reference page in the *OSF DCE Administration Reference*.)

To create the script, you need to gather the following information about the diskless clients and the swap server hosts. These questions are addressed in the sections that follow.

- For each diskless client:
  - — What is the maximum swap space that is needed for all users?
  - — What is the minimum swap space that is needed?
- For each swap server:
  - — How much mass storage can it devote to swap space for diskless clients?
  - — What local devices are available (typically, disk storage)?
  - — What local file space is available in the file system?
- For the cell as a whole:
  - — What is the total space that is available on all swap servers?
  - — What is the maximum space that is needed by all diskless clients?
  - — How many possible users are there on the clients?
  - — How can the diskless clients be distributed across the swap servers to maximize throughput and balance the load?

26.1 The Diskless Client

The parameters that are affected are as follows:

- The **-max** *mK* parameter of the **dsw_adm addclient** command
- The **-init** *iK* parameter of the **dsw_adm addclient** command

To determine the swap space requirements of a diskless client host and its operating system, consult the documentation that is supplied by the vendor who provided the equipment and software.

The maximum swap space (**-max** parameter) is the most space a client can be allocated on the server. It should be able to handle the needs of the largest processes that users can execute on the client host.

The minimum swap space (**-init** parameter) is the least space a client can be allocated on the server. It should be able to handle any operating system requirements and other minimums that are required by user programs.

A static client is one that requests all of its swap space when it initializes. If its configured maximum space is not available, the client is granted a smaller amount, down to its configured minimum. If the minimum is not available, the initialization is denied.

A dynamic client is one that can make allocation requests after it initializes, up to its configured maximum space. It is allocated its configured minimum swap space when it initializes. If the minimum is not available, the initialization is denied.

A client indicates whether it is static or dynamic in its initialization request to the swap server.

26.2 The Swap Server

The parameters that are affected are as follows:

- The **-file** *f* parameter of the **dsw_adm addfile** command
- The **-size** *sK* parameter of the **dsw_adm addfile** command
- The **-file** *ff* parameter of the **dsw_adm addclient** command

Determine what space you have available on each swap server. Make a list of the filenames and the sizes of the devices and the ordinary files you select.

You can use all or part of the device files (usually, disk drives), as well as the ordinary files from the file system. These files must be local to the server host. The swap space that is managed by the diskless swap server is separate and distinct from the space that the server host uses for its own swap area.

Direct access storage devices, such as disk drives, are the preferred media for swapping. These devices must be known to the host; they can be shared. Ordinary files are useful for temporary extensions while awaiting disk upgrades.

26.3 Clients and Servers Together

Having determined the needs of the diskless clients and the resources of the intended swap servers, your next step is to apportion the clients among the servers.

The factors to consider are as follows:

- The number of simultaneous users that are supported by each client
- The volume of swapping that is required by each client, in terms of both frequency and quantity
- The amount of other activity support that is required by each swap server

The goal is to distribute the diskless clients across the swap servers to maximize throughput and balance the load.

When you reach this point in setting up your swap servers, you will have the following:

- A list of swap server hosts
- For each server host, a list of device files or ordinary files or both, along with the amount of space that is available on each one
- For each server host, a list of the client hosts that are to be supported
- For each client host, the maximum and minimum amount of swap space that it requires
- Optionally, for each client host, a list of the server files that it is restricted to

26.4 Creating the Configuration Script

The next step is to create a shell script that contains the appropriate commands to start the swap server daemon and configure the server. When it is completed and tested, incorporate it into the initialization routines that run on the server host when the host is booted. The configuration script contains the following commands:

- A **dswd** command to start the server daemon
- The **dsw_adm addfile** commands to configure the individual swap files
- The **dsw_adm addclient** commands to configure the individual diskless clients

Note: The compiled maximums in the **dswd** daemon are 32 files and 32 clients.

The commands are described in detail in the **dsw_adm(8dskl)** and **dswd(8dskl)** reference pages in the *OSF DCE Administration Reference*. In the examples that follow, it is assumed that the script is executed on and for a server whose hostname is **swaphost1**.

26.4.1 The dswd Command

The **dswd** command is required to start the **dswd** swap server daemon on the swap server host. It must be the first command in the script. It must be executed on the swap server host, which is **swaphost1** in our example. The basic command is as follows:

```
dswd
```

If you expect this server to be heavily used for swapping, and the server host processes program threads efficiently, you may be able to improve

throughput by specifying a larger number of threads for the daemon, as in the following syntax:

dswd -numthreads *n*

where *n* is a positive integer that specifies the number of simultaneous threads that can serve requests from **dsw_adm** commands or diskless clients. The default is 4.

26.4.2 The dsw_adm addfile Command

You need one **dsw_adm addfile** command for each device file or ordinary file that forms the swap space for the server. These are the files that you selected in the preceding sections of this chapter.

26.4.2.1 Adding a Device

To add a device to the swap server configuration, the command syntax is as follows:

dsw_adm addfile -server *s* **-file** *f*

where *s* is the name of the swap server host, and *f* is the filename of the device. For example,

dsw_adm addfile -server swaphost1 -file /dev/dsk/c0d5s3

configures the disk whose device filename is **/dev/dsk/c0d5s3** as a swap file on our sample server that is named **swaphost1**.

Note: If the operating system on the swap server host cannot determine the total space on a device, you must specify the **-size** parameter. Use the syntax shown in Section 26.4.2.2.

26.4.2.2 Adding Part of a Device

If you want to share a device with the file system or some other program, or if the operating system on the swap server host cannot determine the total space on a device, you can specify the number of 1-kilobyte blocks that you want to use for the swap area by adding the **-size** *sK* parameter. The swap server always uses the storage locations starting at byte address 0 (zero). For example,

dsw_adm addfile -server swaphost1 -file /dev/dsk/c0d5s3 -size 40960

configures the first 40 megabytes of the disk for the swap area, starting at byte 0 (zero). When using the balance of the disk, you must be careful to specify addresses that are above the configured area, in this case, at byte address 41943040 or above.

Note: One KB (kilobyte) = 1024 8-bit bytes. One MB (megabyte) = 1024KB = 1,048,576 bytes.

26.4.2.3 Adding an Ordinary File

To add a file system file to the swap server configuration, the command syntax is as follows:

dsw_adm addfile -server *s* **-file** *f* **-size** *sK*

where *s* is the name of the swap server host, *f* is the pathname of the file, and *sK* is the size of the file in kilobytes. The **-size** parameter is required in this case. When the file is added, the **dswd** daemon creates it, if necessary, and preallocates its length to the size requested. For example,

dsw_adm addfile -server swaphost1 -file /other/swapfile2 -size 4096

configures the file whose pathname is **/other/swapfile2** as a swap file in the file system of our sample server host, **swaphost1**. The file length is preallocated to 4 megabytes.

26.4.2.4 Setting Priority

Normally, the swap server determines the most efficient way to use the configured swap space. However, there may be certain files that you would prefer to use only as a last resort, perhaps because access to them is relatively slow. You can prioritize this use by adding the **-priority** *p* parameter to any of the preceding commands. The value of *p* can range from 1 to 9. The default is 1. Files with lower numbers are used first. For example,

dsw_adm addfile -server swaphost1 -file /dev/dsk/c0d5s3 -priority 2

configures the disk whose device name is **/dev/dsk/c0d5s3** as a swap file with priority **2**. It is used only after all of the space on files with priority 1 has been used up.

This feature can also be used to reserve a specific file for use by a particular client. The **dsw_adm addclient** command has a parameter that can restrict a client to certain swap files. (See Section 26.4.3.2 for more information.) By setting the priority of those files to a high number, such as 9, you prevent unrestricted clients from allocating space on those files until space runs out on lower priority files.

26.4.3 The dsw_adm addclient Command

You need one **dsw_adm addclient** command for each diskless client that uses a swap server. These are the clients that you selected in the preceding sections of this chapter.

26.4.3.1 Adding a Client

To add a client to the swap server configuration, the command format is as follows:

dsw_adm addclient -server *s* **-client** *c* **-init** *iK* **-max** *mK*

where *s* is the name of the swap server host, *c* is the hostname of the diskless client, *iK* is the initial or minimum amount of swap area, in

kilobytes, that can be allocated to the client, and *mK* is the maximum amount of swap area, in kilobytes, that can be allocated to the client. For example,

dsw_adm addclient -server swaphost1 -client diskless3 -init 4096 -max 40960

adds client **diskless3** to the configuration of swap server **swaphost1**. If the client is the dynamic type, it is allocated 4 megabytes of space when it requests initialization. It can request additional amounts of space up to 40 megabytes. If the client is the static type, it is allocated 40 megabytes of space when it requests initialization. If the maximum amount of space is not available, it is allocated a lesser amount, down to a minimum of 4 megabytes.

26.4.3.2 Restricting a Client to Certain Files

If you want a client to swap only to specific swap files, you can add the **-file** *ff* parameter to the **dsw_adm addclient** command. The variable *ff* can be one or more *previously configured* swap filenames, separated by spaces. For example,

**dsw_adm addclient -server swaphost1 -client diskless3 -init 4096 -max 40960 **
-file /dev/dsk/c0d5s3 /other/swapfile2

adds client **diskless3** to the configuration of swap server **swaphost1**, and restricts its swap file access to the device named **/dev/dsk/c0d5s3** and the file system file named **/other/swapfile2**. The \ (backslash) at the end of the first line indicates line continuation to the shell command processor.

(See Section 26.4.2.4 for a description of how to deter the allocation of space on these files to other clients.)

26.5 A Sample Configuration Script

The following file combines the examples that are described in the preceding sections into a script that starts and configures a swap server on host **swaphost1**. Information about static or dynamic client types is provided separately by the vendor of the corresponding client operating system. Lines beginning with the # (number sign) character are comments that are ignored by the command processor.

```
#          Be sure the rpcd daemon is running before executing this script.
dswd
#          Starts the swap server daemon on the host.

dsw_adm addfile -server swaphost1 -file /dev/dsk/c0d5s3
#          Entire device is configured as swap space.

dsw_adm addfile -server swaphost1 -file /dev/dsk/c1d3s5 -size 8192 \
        -priority 8
#          8MB of device is configured as swap space.
#          Priority is 8 so unrestricted clients will rarely use it

dsw_adm addfile -server swaphost1 -file /other/swapfile2 -size 4096 \
        -priority 9
#          4MB of ordinary file is configured and preallocated.
#          Priority is 9 so unrestricted clients will rarely use it.

dsw_adm addfile -server swaphost1 -file /dev/dsk/c2d4s6
#          Entire device is configured as swap space.

dsw_adm addclient -server swaphost1 -client diskless1 -init 2048 -max 20480
#          Static client configured with 20MB of swap.
#          It is able to run with a minimum of 2MB.

dsw_adm addclient -server swaphost1 -client diskless3 -init 4096 -max 40960
#          Dynamic client configured, initializes with 4MB of swap,
#          Can be allocated up to 40MB.
```

```
dsw_adm addclient -server swaphost1 -client diskless7 -init 1024 \
        -max 10240 -file /dev/dsk/c1d3s5 /other/swapfile2
#          Dynamic client configured, initializes with 1MB of swap,
#          Can be allocated up to 10MB.
#          Swapping restricted to two files.
#          The disk has lower priority and is used first.
#          The ordinary file is used second if necessary.
#          These files will probably not be used by the other clients.
```

26.6 Changing a Configuration

On occasion, you may want to change a configuration while the host and the swap server continue running. For example, a client may be running out of swap space, resulting in swap error messages on its console or user terminals. Or you may need to take a swap file out of service for some reason. The **dsw_adm** command provides a number of subcommands that let you do the following:

- Add a new swap file to the current configuration (**addfile**)
- Remove a swap file from service (**delfile**)
- Change the priority of a swap file (**modfile**)
- Display the current configuration of the swap files (**examfile**)
- Add a new client to the current configuration (**addclient**)
- Remove a client from access to the server (**delclient**)
- Change the maximum allocation limit of a client (**modclient**)
- Add a file to a client's list of permitted swap files (**modclient**)
- Display the current configuration of the clients (**examclient**)

If you make some changes that you want to keep, remember to modify the start-up script to match.

The **addfile** and **addclient** subcommands are described in detail in Section 26.4. The other subcommands are described in the following subsections.

26.6.1 Removing a Swap File

You can remove a swap file from the configuration of a swap server with the **dsw_adm delfile** command only if no space is allocated on the file to any clients. You can check the allocation with the **dsw_adm examfile** command. You can remove a client's allocation from the file (and remove the client from the swap configuration) with the **dsw_adm delclient** command.

Caution: If the client is active, **delclient** will probably cause the client to abruptly abort its operation.

To remove a swap file from the configuration, the command syntax is as follows:

dsw_adm delfile -server *s* **-file** *f*

For example,

dsw_adm delfile -server swaphost1 -file /dev/dsk/c0d5s3

deletes disk **/dev/dsk/c0d5s3** from the configuration of the swap server on **swaphost1**.

26.6.2 Changing the Priority of a Swap File

You can change the priority of a swap file with the **dsw_adm modfile** command. Changing the priority does not affect any current allocations on the file, only subsequent allocation requests.

To change the priority of a swap file, the command syntax is as follows:

dsw_adm modfile -server *s* **-file** *f* **-priority** *p*

The variable *p* has the range 1 to 9. Files with lower priorities are used first.

For example,

dsw_adm modfile -server swaphost1 -file /dev/dsk/c0d5s3 -priority 4

changes the priority of disk **/dev/dsk/c0d5s3** on swap server **swaphost1** to **4**.

26.6.3 Examining the Swap File Configuration

You can display the current configuration and space allocation of the swap files with the **dsw_adm examfile** command. For each swap file on the server, **examfile** displays the pathname, the total space on the file, the remaining free space, the priority of the file, the clients that currently have space allocated on it, and the space that is allocated to each client. The display is sent to standard output.

To display the file configuration of a server, the command syntax is as follows:

dsw_adm examfile -server *s*

For example,

dsw_adm examfile -server swaphost1

displays the files and the allocated clients for swap server **swaphost1**.

26.6.4 Removing a Client

You can remove a client from the configuration of a swap server with the **dsw_adm delclient** command. Space that is allocated to the client is freed for use by other clients. You can check the allocation with the **dsw_adm examfile** command.

Caution: If the client is active, **delclient** will probably cause the client to abruptly abort its operation.

To remove a client from the configuration, the command syntax is as follows:

dsw_adm delclient -server *s* **-client** *c*

For example,

dsw_adm delclient -server swaphost1 -client diskless7

deletes client **diskless7** from the configuration of the swap server on **swaphost1**.

26.6.5 Increasing a Client's Maximum Allocation

You can increase a client's maximum space allocation with the **-max** *mK* parameter of the **dsw_adm modclient** command. The value of *mK* is given in kilobytes of storage. The new value does not affect any currently allocated space. You cannot decrease the maximum for a currently configured client. A static client can obtain the new maximum the next time it initializes its swap space. A dynamic client can use the increased space immediately.

To change the maximum allocation for a client, the command syntax is as follows:

dsw_adm modclient -server *s* **-client** *c* **-max** *mK*

For example,

dsw_adm modclient -server swaphost1 -client diskless3 -max 20480

changes the maximum allocation for client **diskless3** on server **swaphost1** to 20 megabytes.

26.6.6 Adding Files to a Client's Permitted Swap Files

You can add one or more files to the list of files where a client is permitted to swap with the **-file** *ff* parameter of the **dsw_adm modclient** command. The value of *ff* is one or more currently configured swap filenames, separated by spaces. The specified files are added to the client's current list. If no list exists, one is created. You cannot delete files from a list, except by using **delclient** to remove the client entirely and then **addclient** to reconfigure the client.

To add files to a client's list of permitted swap files, the command syntax is as follows:

dsw_adm modclient -server *s* **-client** *c* **-file** *ff*

For example,

```
dsw_adm modclient -server swaphost1 -client diskless3 \
        -file /dev/dsk/c0d5s3 /other/swapfile2
```

adds the files **/dev/dsk/c0d5s3** and **/other/swapfile2** to the permitted swap file list of client **diskless3** on server **swaphost1**.

26.6.7 Examining the Client Configuration

You can display the current configuration of the clients on a server with the **dsw_adm examclient** command. For each client, **examclient** displays the client's hostname, its initial and maximum allocations, and (if configured) its list of permitted swap files. If no files are displayed, the client is not restricted. The display is sent to standard output.

To display the client configuration of a server, the command syntax is as follows:

dsw_adm examclient -server *s*

For example,

```
dsw_adm examclient -server swaphost1
```

displays the clients, their allocation limits, and the permitted files for swap server **swaphost1**.

26.6.8 Getting Help

The **dsw_adm help** command displays an annotated summary of the **dsw_adm** subcommands on standard output. The command syntax is as follows:

dsw_adm help

The **-help** parameter displays a syntax summary for any subcommand. The command syntax is as follows:

dsw_adm *subcommand* **-help**

where *subcommand* is any one of **addclient**, **addfile**, **delclient**, **delfile**, **examclient**, **examfile**, **help**, **modclient**, or **modfile**.

For example,

dsw_adm addfile -help

displays the syntax of the **addfile** subcommand.

26.7 Moving a Client to Another Swap Server

In the normal course of operations, you may decide to move a diskless client from one swap server to another. While the client is only permitted to swap on one server at a time, based on its swap server entry in the DCE Cell Directory Service, more than one swap server can be configured to handle the same client. Consequently, the process of moving a client to another server is accomplished by performing the following steps.

Because the **dsw_adm** command can be executed for any swap server from any host in the cell, most (possibly all) of the steps can be performed from a single host.

The examples in this process show diskless client **diskless1** being transferred from swap server host **oldserver** to swap server host **newserver**.

1. Check the client's current configuration on the current server.

   dsw_adm examclient -server oldserver

2. Add the client to the configuration of the new server host by using the initial and maximum allocation values that are shown in the previous step. These values can be modified if necessary. This step is needed if the new server is not rebooted before the client is ready to reboot.

   dsw_adm addclient -server newserver -client diskless1 -init *oldinit* \
   -max *oldmax*

3. Add the command from the previous step to the start-up script for the new server. Depending on file access, you may have to log in directly to the new server. At this point, you may want to add file restrictions to the **dsw_adm addclient** command.

4. Change the swap server entry for the client in the DCE Cell Directory Service to point to the new swap server. Change the entry in the **dlctab** configuration file and then execute the **dlc_adm update** command to install the change.

   At this point, when the client reboots, it uses the new server.

5. Remove the **addclient** entry for the client from the old server's start-up script.

6. After the client shuts down or reboots and is swapping on the new server, delete the client from the old server.

   dsw_adm delclient -server oldserver -client diskless1

Appendix A

Structure Rule Table

The Structure Rule Table (SRT) supplied with the GDS default schema contains the entries listed in Tables A-1 and A-2.

Table A-1 describes the structure rules defined for a DSA. (See Chapter 9 for more information.)

Table A–1. DSA SRT Entries

Rule Number	Superior Rule Number	Acronyms of Naming Attribute	Acronym of Structural Object Class
1	0	CN	SCH
2	0	C	C
3	2	O	ORG
4	3	OU	OU
5	4	CN	ORP
6	4	CN	ORR
7	4	CN	APP
8	4	CN	MDL
9	4	CN, OU	ORP
10	7	CN	APE
11	7	CN	DSA
12	7	CN	MMS
13	7	CN	MTA
14	7	CN	MUA
15	2	L	LOC
16	15	CN	REP
17	15	CN, STA	REP

Notes:

- See Table C-1 in Appendix C for an explanation of the acronyms of the naming attributes.
- See Table B-1 in Appendix B for an explanation of the acronyms of object classes.

Table A-2 describes the SRT defined for the GDS administration programs.

Table A–2. SRT Entries for GDS Administration Programs

Rule Number	Superior Rule Number	Acronyms of Naming Attribute	Acronym of Structural Object Class
1	0	CN	SCH
2	0	C	C
3	2	O	ORG
4	3	OU	OU
5	4	CN	ORP, APP, ORR, MDL
6	4	CN, OU	ORP
7	5	CN	DSA, APE, MMS, MTA, MUA
8	2	L	LOC
9	15	CN	REP
10	15	CN, STA	REP

Appendix B

Object Class Table

The Object Class Table (OCT) supplied with the GDS default schema contains the entries listed in Table B-1. (See Chapter 9 for more information.)

Table B–1. OCT Entries

Object Class			Aux.	Super-	OID	File	Mandatory	Optional
Acronym	Name	Kind		class		No.	Attributes	Attributes
TOP	Top	Abstract	-	None	85.6.0	-1	OCL	None
ALI	Alias	Alias	-	TOP	85.6.1	-1	AON	None
C	Country	Structural	-	GTP	85.6.2	1	C	DSC SG CDC CDR
LOC	Locality	Structural	-	GTP	85.6.3	4	None	DSC L SPN STA SEA SG CDC CDR
ORG	Organization	Structural	-	GTP	85.6.4	1	O	DSC L SPN STA PDO PA PC POB FTN IIN TN TTI TXN X1A PDM DI RA SEA UP BC SG CDC CDR
OU	Organizational-Unit	Structural	-	GTP	85.6.5	1	OU	DSC L SPN STA PDO PA PC POB FTN IIN TN TTI TXN X1A PDM DI RA SEA UP BC SG CDC CDR
PER	Person	Abstract	-	GTP	85.6.6	-1	CN SN	DSC TN SEA UP CDC CDR

Object Class			Aux.	Super-class	OID	File No.	Mandatory Attributes	Optional Attributes
Acronym	Name	Kind						
ORP	Organizational-Person	Structural	MUS	PER	85.6.7	1	None	L SPN STA PDO PA PC POB FTN IIN TTI TXN X1A PDM DI RA OU TIT
ORR	Organizational-Role	Structural	-	GTP	85.6.8	1	CN	DSC L SPN STA PDO PA PC POB FTN IIN TN TTI TXN X1A PDM DI RA OU SEA RO CDC CDR
GON	Group-of-Names	Structural	-	GTP	85.6.9	3	CN MEM	DSC O OU SEA BC OWN CDC CDR
REP	Residential-Person	Structural	-	PER	85.6.10	4	L	SPN STA PDO PA PC POB FTN IIN TTI TXN X1A PDM DI RA BC
APP	Application-Process	Structural	-	GTP	85.6.11	2	CN	DSC L OU SEA CDC CDR

Object Class			Aux.	Super-	OID	File	Mandatory	Optional
Acronym	Name	Kind		class		No.	Attributes	Attributes
APE	Application-Entity	Structural	-	GTP	85.6.12	2	CN PSA	DSC L O OU SEA SAC CDC CDR
DSA	Directory-Service-Agent	Structural	-	APE	85.6.13	2	None	KNI UP
DEV	Device	Structural	-	GTP	85.6.14	2	CN	DSC L O OU SEA OWN SER CDC CDR
MDL	MHS-Distribution-List	Structural	-	GTP	86.5.1.0	2	CN MDS MOA	DSC O OU SEA OWN MDT MDE MDM MPD
MMS	MHS-Message-Store	Structural	-	APE	86.5.1.1	2	None	OWN MSO MSA MSC
MTA	MHS-Mess-Transfer-Agent	Structural	-	APE	86.5.1.2	2	None	OWN MDL
MUS	MHS-User	Auxiliary	-	TOP	86.5.1.3	-1	MOA	MDL MDT MDE MMS MPD
MUA	MHS-User-Agent	Structural	-	APE	86.5.1.4	2	None	OWN MDL MDT MDE MOA
SCH	Schema	Structural	-	GTP	43.12.2. 1107.1.3. 6.0	0	CN	TST SRT OCT AT

Object Class			Aux.	Super-	OID	File	Mandatory	Optional
Acronym	Name	Kind		class		No.	Attributes	Attributes
GTP	GDS-Top	Abstract	-	TOP	None	-1	None	ACL MK

See Table C-1 in Appendix C for an explanation of acronyms of the mandatory attributes and the optional attributes.

Notes: Although the object class **Locality (LOC)** does not have a specific set of mandatory attributes, either the **Locality-Name (L)** or the **State-or-Province-Name (SPN)** attribute must be present. (See Appendix C for more information.)

Every object class inherits all mandatory and optional attributes of its superior object classes.

The object classes **Group-of-Names (GON)** and **Device (DEV)** are object classes defined in the X.500 standard. They are part of the OCT, but they are not assigned to any structure rule of the SRT of the GDS default schema. They can be used in schema administration functions if an administrator creates two new structure rules.

Appendix C

Attribute Table

The Attribute Table (AT) supplied with the GDS default schema contains the entries described in Table C-1. (See Chapter 9 for more information.)

Table C–1. AT Entries

Attribute									
Acr.	Name	OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
OCL	Object-Class	85.4.0	1	28	0	2	0	0	0
AON	Aliased-Object-Name	85.4.1	1	1024	1	1	0	0	0
KNI	Knowledge-Information	85.4.2	1	1024	0	4	0	0	0
CN	Common-Name	85.4.3	1	64	2	4	1	0	1
SN	Surname	85.4.4	1	64	2	4	1	0	0
SER	Serial-Number	85.4.5	1	64	2	5	0	0	0
C	Country-Name	85.4.6	2	2	1	1010	1	0	1
L	Locality-Name	85.4.7	1	128	2	4	1	0	1
SPN	State-or-Province-Name	85.4.8	1	128	2	4	1	0	0
STA	Street-Address	85.4.9	1	128	2	4	1	0	0
O	Organization-Name	85.4.10	1	64	2	4	1	0	1
OU	Org.-Unit-Name	85.4.11	1	64	2	4	1	0	1
TIT	Title	85.4.12	1	64	2	4	1	0	0
DSC	Description	85.4.13	1	1024	0	4	0	0	0
SG	Search-Guide	85.4.14	1	256	0	1000	0	0	0
BC	Business-Category	85.4.15	1	128	2	4	1	0	0
PA	Postal-Address	85.4.16	1	180	2	1001	1	1	0
PC	Postal-Code	85.4.17	1	40	2	4	1	0	0
POB	Post-Office-Box	85.4.18	1	40	2	4	1	0	0
PDO	Phys.-Deliv.-Office-Name	85.4.19	1	128	2	4	1	0	0
TN	Telephone-Number	85.4.20	1	32	0	12	0	0	0

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	Name								
TXN	Telex-Number	85.4.21	1	26	0	1002	0	0	0
TTI	TTX-Terminal-Identifier	85.4.22	1	1024	0	1003	0	0	0
FTN	Fax-Telephone-Number	85.4.23	1	37	0	1004	0	0	0
X1A	X121-Address	85.4.24	1	15	0	6	0	0	0
IIN	Internat.-ISDN-Number	85.4.25	1	16	0	6	0	0	0
RA	Registered-Address	85.4.26	1	180	2	1001	1	0	0
DI	Destination-Indicator	85.4.27	1	128	2	4	1	0	0
PDM	Preferred-Delivery-Method	85.4.28	0	10	1	1005	0	0	0
PSA	Presentation-Address	85.4.29	1	268	1	1006	0	0	0
SAC	Suppl.-Applic.-Context	85.4.30	1	28	2	2	0	0	0
MEM	Member	85.4.31	1	1024	0	1	0	0	0
OWN	Owner	85.4.32	1	1024	0	1	0	0	0
RO	Role-Occupant	85.4.33	1	1024	0	1	0	0	0
SEA	See-Also	85.4.34	1	1024	0	1	0	0	0
UP	User-Password	85.4.35	0	128	2	1011	0	2	0
MDL	MHS-Deliv.-Cont.-Lengthl	86.5.2.0	4	4	1	9	0	0	0
MDT	MHS-Deliv.-Cont.-Types	86.5.2.1	1	28	4	2	0	0	0
MDE	MHS-Deliverable-EITs	86.5.2.2	1	28	8	2	0	0	0
MDM	MHS-DL-Members	86.5.2.3	1	3596	0	102	0	0	0
MDS	MHS-DL-Submit-								

Attribute									
Acr.	Name	OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
	Permissions	86.5.2.4	1	3604	0	100	0	0	0
MMS	MHS-Message-Store	86.5.2.5	1	1024	1	1	0	0	0
MOA	MHS-O/R-Address	86.5.2.6	1	2564	0	101	0	0	0
MPD	MHS-Pref.-Deliv.-Meth.	86.5.2.7	0	10	1	103	0	0	0
MSA	MHS-Supp.-Autom.-Action	86.5.2.8	1	28	4	2	0	0	0
MSC	MHS-Supp.-Content-Types	86.5.2.9	1	28	4	2	0	0	0
MSO	MHS-Supp.-Optional-Attr.	86.5.2.10	1	28	0	2	0	0	0
MK	Master-Knowledge	43.12.2.11 07.1.3.4	1	1024	1	1	0	0	0
ACL	Access-Control-List	43.12.2.11 07.1.3.4.1	1	20500	1	10000	0	0	0
TST	Time-Stamp	43.12.2.11 07.1.3.4.2	11	18	1	11	0	0	0
SRT	Structure-Rule-Table	43.12.2.11 07.1.3.4.4	1	29	0	5	0	0	0
OCT	Object-Class-Table	43.12.2.11 07.1.3.4.5	1	397	0	5	0	0	0
AT	Attribute-Table	43.12.2.11 07.1.3.4.6	1	101	0	5	0	0	0
CDC	CDS-Cell	43.12.2.11 07.1.3.4.13	1	284	1	10	0	0	0
CDR	CDS-Replica	43.12.2.11 07.1.3.4.14	1	905	0	10	0	0	0

C.1 Explanations

Tables C-2 through C-5 provide explanations of the syntaxes, phonetic flags, maximum number of values, and access classes used in Table C-1.

Table C–2. Syntaxes

Digit	Syntax
0	Any
1	Distinguished Name
2	Object Identifier
3	Case Exact String
4	Case Ignore String
5	Printable String
6	Numeric String
7	Case Ignore List
8	Boolean
9	Integer
10	Octet String
11	UTC Time
12	Telephone Number
100	MHS DL Submit Permission
101	MHS O/R Address
102	MHS O/R Name
103	MHS Preferred Delivery Method
1000	Search Guide
1001	Postal Address
1002	Telex Number
1003	Teletex Terminal Identifier
1004	Fax Number
1005	Preferred Delivery Method
1006	Presentation Address
1010	Country Name
1011	Password
10000	Access Control List

Table C–3. Phonetic Flags

Digit	Flag
0	No phonetic matching
1	Phonetic matching

Table C–4. Access Classes

Digit	Access Class
0	Public
1	Standard
2	Sensitive

Table C–5. Maximum Number of Values

Digit	Explanation
0	Unlimited number of values
>0	Upper bound for number of values

Appendix D

PSAP Addresses

This appendix provides an administrator with all the necessary information to enter or display PSAP addresses using the GDS administration program **gdsditadm(8gds)** or **gdscacheadm(8gds)**. The syntaxes of the different components of a PSAP address are described in Backus Naur Form (BNF) notation.

Note: In order to improve legibility, the keyword OR is used instead of the | (pipe) symbol.

D.1 Examples of Entering Client and Server Addresses

PSAP addresses are entered in Mask 7a of the administration programs **gdsditadm** or **gdscacheadm**. Figure D-1 shows how a client address is entered in Mask 7a.

Figure D-2 shows how a server address is entered in Mask 7a.

Figure D–1. Example of a Client Address

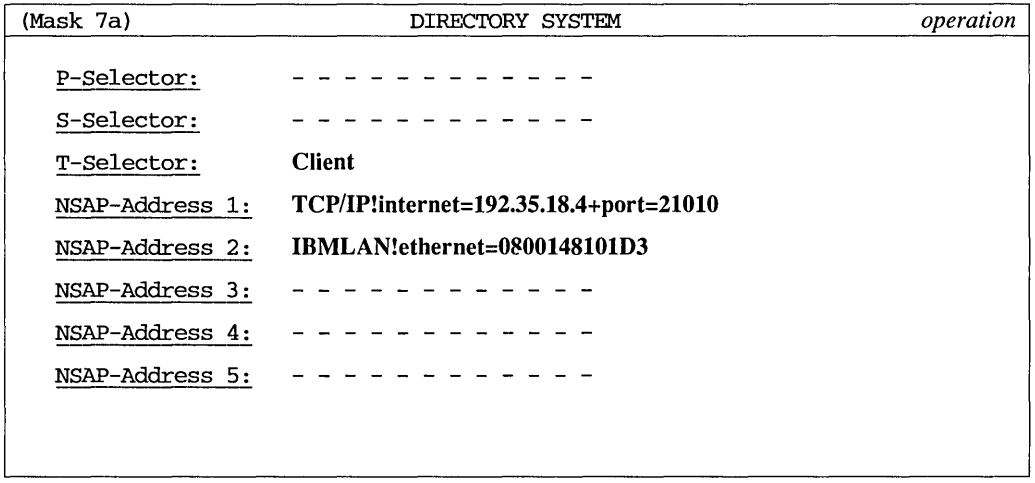
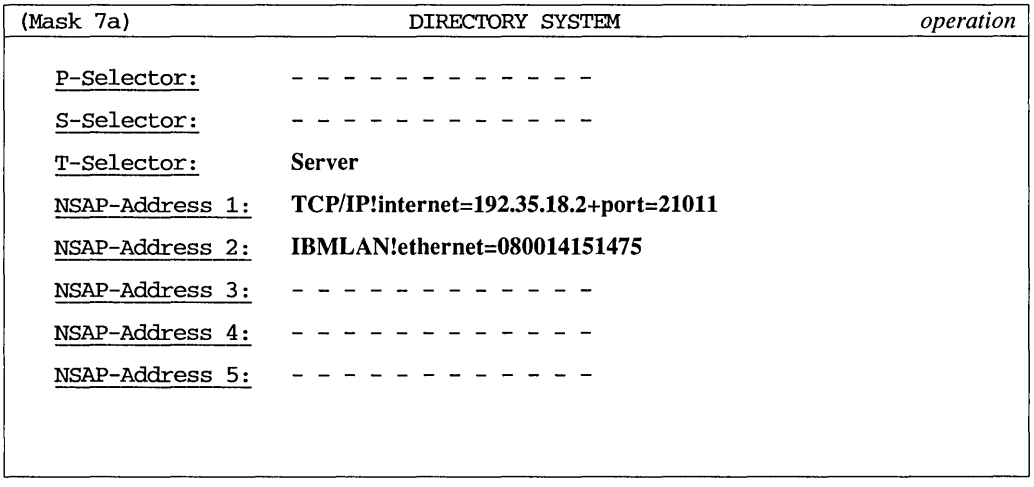


Figure D–2. Example of a Server Address



Figures D-1 and D-2 show two types of NSAP address:

NSAP-Address 1
: This is the socket address. Administrators must ensure that the port numbers (entered with **port=21010** or **port=21011** in Figures D-1 and D-2) are unique on their machines.

NSAP-Address 2
: This is the LAN address used on RS6000 IBM machines.

The addresses are always administered as indicated in the figures.

GDS operates XTI as well as sockets. The environment variable **TSITYPE** determines which network is used. If **TSITYPE** is **SOCKET**, only addresses of the **NSAP-address 1** type need to be administered.

NSAP-address 2 is an example of a typical XTI address.

If the administrator wants to use both address types in an installation, both addresses can be stored in the DUA cache or DIT for the same object. The network type used depends on the value of the environment variable **TSITYPE**.

If the administrator wants to specify other address types for XTI, the **nsapform.cfg** file (see Section D.3.1) and the **nsapmacros** file (see Section D.4) must be modified.

The remainder of this appendix provides a detailed description of all other information concerning PSAP addresses.

D.2 Presentation/Session/Transport Selector Syntax

This section deals with the presentation, session, and transport selector syntax portions of a PSAP address.

```
<selector> ::=
        <anystring><subset any>
    OR
        <anystring>'
    OR
        x'<hexstring>'    1)
```

Note: If the syntax of the frame x'<...>' is specified incorrectly, then the selector value is interpreted as **<anystring>'** or **<anystring><subset any>**.

```
<anystring> ::=
        <any>
    OR
        <any><anystring>


<any> ::= [printable ASCII-character set (\040 - \176)]


<subset any> ::= [printable ASCII-character set
                        (\041 - \053, \055 - \176)]


<hexstring> ::=
        <hexvalue>
    OR
        <hexvalue><hexstring>


<hexvalue> ::= [0-9a-fA-F][0-9a-fA-F]
```

Examples for a T-Selector
Server for the T-Selector of a server
Client for the T-Selector of a client

D.3 Common NSAP Address Syntax

This section shows the common NSAP address syntax.

```
<NSAP-address> ::=
      <NSAP-address type>+<IDI-value>
   OR
      <NSAP-address type>+<DSP>
   OR
```

```
      <NSAP-address type>+<IDI-value>+<DSP>
   OR
       NA+<hexstring>


<NSAP-address type> ::=  1)
      X121_D               /* CCITT-X.121 address type
                                 (decimal DSP-syntax) */
   OR
      X121_B               /* CCITT-X.121 address type
                                 (binary DSP-syntax)  */
   OR
      DCC_D                /* ISO-3166-DCC address type
                                 (decimal DSP-syntax) */
   OR
      DCC_B                /* ISO-3166-DCC address type
                                 (binary DSP-syntax)  */
   OR
      TELEX_D              /* CCITT-F.69 address type
                                 (decimal DSP-syntax) */
   OR
      TELEX_B              /* CCITT-F.69 address type
                                 (binary DSP-syntax)  */
   OR
      PSTN_D               /* CCITT-E.163 address type
                                 (decimal DSP-syntax) */
   OR
      PSTN_B               /* CCITT-E.163 address type
                                 (binary DSP-syntax)  */
   OR
      ISDN_D               /* CCITT-E.164 address type
                                 (decimal DSP-syntax) */
   OR
      ISDN_B               /* CCITT-E.164 address type
                                 (binary DSP-syntax)  */
   OR
      ICD_D                /* ISO-6523-ICD address type
                                 (decimal DSP-syntax) */
   OR
      ICD_B                /* ISO-6523-ICD address type
                                 (binary DSP-syntax)  */
```

```
OR
   LOCAL_D              /* local address type
                              (decimal DSP-syntax) */
OR
   LOCAL_B              /* local address type
                              (binary DSP-syntax)  */
OR
   LOC_ISO              /* local address type
                              (ISO646 DSP-syntax)  */
OR
   LOC_NAT              /* local address type
                              (national DSP-syntax)*/
```

D.3.1 OSI-NSAP Address Format Description Table

The example in this section describes the different NSAP address types with reference to ISO 8348 Addendum 2, and, in particular, the DSP formats that are supported by the communications software. Each NSAP address type supported can have one or more entries, where each entry consists of the following:

- The name of the NSAP address type (only predefined names can be used) or the redirection symbol > (only if the entry is an additional entry belonging to the same NSAP address type).

  The following values are optional for each entry:

  — An Initial Domain Identifier (IDI) for which the following syntaxes are permitted:

  IDI-value The DSP format that follows is associated with the given IDI value only.

  IDI-value1-IDI-value2
  The DSP format that follows is associated with the given range of IDI values.

  * The DSP format that follows is associated with all IDI values.

+ The DSP format that follows is associated with all IDI values left by other syntaxes.

— A name of the DSP format (up to 8 characters permitted)

— The number of DSP components (up to **MAX_DSP_PARTS**)

— The size of each DSP component; that is, the number of digits (if the DSP syntax is decimal), or the number of octets (if the DSP syntax is binary, ISO646 or national). Together with the first DSP component, a DSP format identifier can be specified (optional). Thus, different DSP structures can be defined in one domain. The syntax is as follows:

size-of-the-first-DSP-component[**:***DSP-format-identifier*]

If a DSP-format identifier is defined, then the maximum size of the first DSP-component can be up to **MAX_DSPID** digits (if the DSP syntax is decimal), or up to **MAX_DSPID/2** octets (if the DSP syntax is binary, ISO646 or national).

Note: Lines beginning with the # (number sign) are comment lines.

Example:

```
#
# FORMAT=CCITT-X.121;AFI=36,52;IDI-length=14,DSP syntax=decimal;\
# DSP=length=24
#
  X121_D

#
# FORMAT=CCITT-X.121;AFI=37,53;IDI-length=14;DSP syntax=binary;\
# DSP-length=12
#

  X121_B

#
# FORMAT=ISO-DCC;AFI=38;IDI-length=3;DSP-syntax=decimal;\
# DSP-length=35
#
# DCC_D
#
```

```
# FORMAT=ISO-DCC;AFI=39;IDI-length=3;DSP-syntax=binary;\
# DSP-length=17
#
# DCC_B
#
# FORMAT=CCITT-F.69;AFI=40,54;IDI-length=8;DSP-syntax=decimal;\
# DSP-length=30
#
  TELEX_D   00728722     QUIPU   4   2:5   12   5   5
  <         00728722     INTX25  3   2:1   14  12
#
# FORMAT=CCITT-F.69;AFI=41,55;IDI-length=8;DSP-syntax=binary;\
# DSP-length=15
#
# TELEX_B
#
# FORMAT=CCITT-E.163;AFI=42,56;IDI-length=12;DSP-syntax=decimal;\
# DSP-length=26
#
# PSTN_D
#
# FORMAT=CCITT-E.164;AFI=44,58;IDI-length=15;DSP-syntax=decimal;\
# DSP-length=23
#
# ISDN_D
#
# FORMAT=CCITT-E.164;AFI=45,59;IDI-length=15;DSP-syntax=binary;\
# DSP-length=11
#
# ISDN_B
#
# FORMAT=ISO-ICD;AFI=46;IDI-length=4;DSP-syntax=decimal;\
# DSP-length=34
#
# ICD_D
#
# FORMAT=ISO-ICD;AFI=47;IDI-length=4;DSP-syntax=binary;\
# DSP-length=17
#
  ICD_B     4-6             GOSIP   4   2   2   6   1
```

```
#
# FORMAT=Local;AFI=48;IDI-length=0;DSP-syntax=decimal;\
# DSP-length=38
#
# LOCAL_D
#
# FORMAT=Local;AFI=49;IDI-length=0;DSP-syntax=binary;\
# DSP-length=19
#
  LOCAL_B   *              GOSIP   4   2   2   6   1
#
# FORMAT=Local;AFI=50;IDI-length=0;DSP-syntax=ISO646;\
# DSP-length=19
#
# LOC_ISO
#
# FORMAT=Local;AFI=51;IDI-length=0;DSP-syntax=National;\
# DSP-length=19
#
# LOC_NAT
```

The preceding example, which corresponds to an IBM/RS6000 machine, is contained in the ASCII file **nsapform.ibm**. The communications software deals with a binary representation of this table contained in the file **nsapform.cfg**. It is for this reason that, each time a change is made, the ASCII file must be converted into the binary file **nsapform.cfg** by means of the tool **gdsgennsap** (syntax: **gdsgennsap** *ASCII-file-name binary-file-name*).

```
<IDI-value> ::= <digitstring>


<DSP> ::=
          <DSP-type>+<decimal DSP-parts>
       OR
          <DSP-type>+<binary DSP-parts>


<DSP-Type> ::=     1)
         QUIPU      /* QUIPU DSP-format  */
     OR
```

```
        GOSIP      /* GOSIP DSP-format  */  2)
    OR
        ECMA117    /* ECMA-117 DSP-format*/ 3)
    OR
        NETBIOS    /* NETBIOS DSP-format */ 3)
```

Notes: The names of DSP types should correspond to the names specified in the NSAP address format description table (see the **nsapform.cfg** or **nsapform.xxx** file in Section D.3).

DSP type not supported on SNI machines.

DSP type not supported on IBM/RS6000 machines.

```
<decimal DSP-parts> ::=
          <digitstrings>
      OR
          <digitstrings>+<decimal DSP-parts>


<binary DSP-parts> ::=
           <hexstrings>
      OR
           <hexstrings>+<binary DSP-parts>


<digitstrings> ::=
           <digitstring>
      OR
           <dotted digitstring>
      OR
           <special ASCII-string>

<hexstrings> ::=
           <hexstring>
      OR
           <dotted digitstring>
      OR
           <special ASCII-string>


<digitstring> ::=
```

```
        <digit value>
    OR
        <digit value><digitstring>


<dotted digitstring> ::=
        .<special digitstring>
    OR
        <special digitstring><multi dotted digitstring>
```

Note: A dotted digit string is internally mapped into a **<hexstring>** (if the DSP syntax is binary; for example, 127.0.0.1 ---> 7F000001) or into a **<triple digitstring>** (if the DSP syntax is decimal; for example, 127.0.0.1 ---> 127000000001).

```
<multi dotted digitstring> ::=
        .<special digitstring>
    OR
        .<special digitstring><multi dotted digitstring>


<special ASCII-string> ::=
       '<subset anystring0>'

<subset anystring0> ::=
        <any0>
    OR
        <any0><subset anystring0>

<any0> ::= [printable ASCII-character set (\040 - \046,
           \050 - \052, \054 - \176)]
```

Note: A special ASCII string is internally mapped into a **<hexstring>** (if the DSP syntax is binary; for example, 'NAME1' ---> 4E414D4531) or into a **<triple digitstring>** (if the DSP-syntax is decimal; for example, 'name1' ---> 110097109101049).

```
<hexstring> ::=
          <hexvalue>
```

```
        OR
             <hexvalue><hexstring>


<digit value> ::= [0-9]


<special digitstring> ::= [0 - 255]


<triple digitstring> ::=
          [000 - 255]
        OR
          [000 - 255]<triple digitstring>


<hexvalue> ::= [0-9a-fA-F][0-9a-fA-F]
```

D.3.2 Concrete DSP Syntaxes

This section shows the concrete DSP syntaxes.

```
<decimal ECMA-DSP> ::=
         ECMA117+<subnet-id/D>+<subnet-address/D>+<N-selector/D>


<binary ECMA-DSP> ::=
        ECM117+<subnet-id/B>+<subnet-address/B>+<N-selector/B>


<decimal QUIPU-DSP> ::=
         QUIPU+<prefix>+<internet-address>+
                                  <port-number>+<transport-set>


<binary NETBIOS-DSP> ::=
        <NETBIOS+<subnet-id/B>+<machine name/B>
```

```
<binary GOSIP-DSP> ::=
        GOSIP+<org-id/B>+<subnet-id/B>+<subnet-address/B>+
                                                <N-selector/B>


<binary GOSIP2-DSP> ::=
        GOSIP2+<org-id/B>+<subnet-id/B>+<subnet-address/B>+
                                                  <N-selector/B>


<subnet-id/D> ::= <digitstring>

<subnet-address/D> ::= <digitstring>

<N-selector/D> ::= <digitstring>

<org-id/B> ::= <hexstring>

<subnet-id/B> ::= <hexstring>

<subnet-address/B> ::= <hexstring>

<N-selector/B> ::= <hexstring>

<prefix> ::= <digitstring>

<internet-address> ::=
         <triple digitstring>
      OR
         <dotted digitstring>

<port-number> ::= <digitstring>

<transport-set> ::= <digitstring>

<machine name/B> ::= '<special ASCII-string>'
```

D.4 Macro Facility

Defining macros simplifies the task of entering NSAP addresses. There are two types of macros; the first type takes the form **<macro name>=<macro value>** and the second type takes the form **<macro name>macro value/macro parameter list>** . Macros are considered by the administration program in both directions (screen input/output). Therefore, the first macro type is interpreted as follows:

```
*Macro type "<macro name>=<macro value>"
```

- Reading NSAP address information from the screen:

  The specified input string is parsed and every (sub)string matching **<macro name>** is substituted by **<macro value>**.

- Writing NSAP information to the screen:

  The specified output string is parsed and every (sub)string matching **<macro value>** is substituted **<macro name>**.

Note: Macros of this type are substituted recursively (see the following example).

The second macro type is interpreted as follows:

```
*Macro type "<macro name>!<macro value/ macro
                                parameter list>"
```

- Reading NSAP address information from the screen

  The specified input string is parsed for **<macro name>**. If there is a macro of this type, then the defined value of **<macro value/macro parameter list>** is used and every parameter name contained in the macro parameter list is substituted by the corresponding macro parameter value from the input string (specified in the format **<macro parameter name>=<macro parameter value>**).

  Note: The macro parameters can be given in the input string in any order.

- Writing NSAP information to the screen:

  The given output string is parsed and, if it matches any macro value **<macro value/macro parameter list>** of the defined macros, then it is substituted by the macro name **<macro name>** and the macro

parameter or parameters **<macro parameter name>=<macro parameter value>**.

Note: To avoid problems, macros of this type need to be defined definitively.

Output strings are first parsed for macros of the second type. They are only parsed for macros of the first type if there are no matching strings of the second type.

Macros can be defined by writing them (one macro per line) into the macro file **nsapmacros**. The contents of the macro file are loaded by the administration program on the first selection of a PSAP handling mask each time the program is called. Both types of macros can be defined in the macro file simultaneously. However, mixing both macro types (using one macro type within a macro of the other type) is not recommended because the results are unpredictable.

D.4.1 NSAP Address Macro Definition Syntax

This section shows the NSAP address macro definition syntax.

```
<NSAP-macro> ::=
        <macro name>=<macro value>
    OR
        <macro name>!<macro parameter list definition>


<macro name> ::= <subset anystring1>


<macro value> ::= <anystring>


<macro parameter list definition> ::=
        <macro parameter name>=
    OR
        <macro parameter name>=+<macro parameter
                                        list definition>
    OR
        <macro parameter name>#+<macro parameter
```

```
                                              list definition>
        OR
           <macro parameter name>@+<macro parameter
                                               list definition>
        OR
           <subset anystring2>
        OR
           <subset anystring2>+<macro parameter
                                              list definition>
```

Note: When # (number sign) or @ (at sign) is specified following a macro parameter name (instead of = (equal sign)), the corresponding macro parameter value is printed in a dotted digit string representation or in a special ASCII string representation.

```
<macro parameter name> ::= <subset anystring3>


<anystring> ::=
         <any>
      OR
         <any><anystring>


<any> ::= [printable ASCII-character set (\040 - \176)]


<subset anystring1> ::=
         <subset any1>
      OR
         <subset any1><subset anystring1>


<subset any1> ::= [printable ASCII-character set
        (\040 - \071, \073 - \074, \076 - \176)]


<subset anystring2> ::=
         <subset any2>
      OR
```

```
        <subset any2><subset anystring2>


<subset any2> ::=
        [printable ASCII-character set
        (\040 - \042, \044 - \074, \076 - \077, \101 - \176)]


<subset anystring3> ::=
        <subset any3>
     OR
        <subset any3><subset anystring3>


<subset any3> ::=
        [printable ASCII-character set
        (\040 - \042, \044 - \052, \054 - \074,
         \076 - \077, \101 - \176)]
```

D.4.2 NSAP Address Macro Calling Syntax

This section shows the NSAP address macro calling syntax.

```
<macro call> ::=
        <macro name>=
     OR
        <macro name>!<macro parameter list>


<macro parameter list> ::=
        <macro parameter name>=<macro parameter value>
     OR
        <macro parameter name>=<macro parameter value>+
                                    <macro parameter list>


        <macro parameter value> ::= <subset anystring3>
```

D.5 Examples of NSAP Addresses

The following are examples of NSAP addresses:

- CCITT-X.121 address, no DSP:

  X121_D+12345678901234

- CCITT-X.121 address, ECMA-117 structured binary DSP:

  X121_B+12345678901234+ECMA117+01+08227619+FE

- CCITT-F.69 address, QUIPU structured decimal DSP:

  TELEX_D+00749044+QUIPU+5+127000000001+4711+0

- Local address, ECMA-117 structured decimal DSP:

  LOCAL_D+ECMA117+123+4567890+1

D.5.1 Examples of Macro Definitions

The following are examples of macro definitions:

TCP/IP!TELEX_D+00728722+QUIPU+5+internet#+port=+2

IBMLAN!ICD_B+4+GOSIP+01+30+ethernet=+01

IBMWAN!X121_D+dte=

NET1!TELEX_B+50093994+NETBIOS+01+name

LOC=LOCAL_D+ECMA117+123+45

X25=X121_D+1234567890

The following shows how the input strings in the preceding macro definitions are expanded:

```
X25=1234                ---> X121_D+12345678901234

LOC=67890+1             ---> LOCAL_D+ECMA117+123+4567890+1

TCP/IP!internet=127.0.0.1+port=4711 --->
            TELEX_D+00749044+QUIPU+5+12700000001+4711+2

NET1!name='machine1'  --->
            TELEX_B+50093994+NETBIOS+01+'machine1'
```

Appendix E

Valid Characters for GDS Naming Attributes

This appendix describes the valid character sets for the GDS naming attributes.

The values of the country attributes are restricted to the ISO 3166 Alpha-2 code representation of country names. These are indicated in the tables in Section E.1.

The character set for all other naming attributes is the T.61 graphical character set. It is described in Section E.2.

E.1 Country Syntax

Country names are represented by a two-letter sequence. GDS does not distinguish between lowercase and uppercase for country names. The complete list of valid combinations is shown in Table E-1 together with the respective names.

Table E–1. Country Syntax

Country Name	Code	Country Name	Code
AFGHANISTAN	AF	ALBANIA	AL
ALGERIA	DZ	AMERICAN SAMOA	AS
ANDORRA	AD	ANGOLA	AO
ANGUILLA	AI	ANTARCTICA	AQ
ANTIGUA AND BARBUDA	AG		
ARGENTINA	AR	ARUBA	AW
AUSTRALIA	AU	AUSTRIA	AT
BAHAMAS	BS	BAHRAIN	BH
BANGLADESH	BD	BARBADOS	BB
BELGIUM	BE	BELIZE	BZ
BENIN	BJ	BERMUDA	BM
BHUTAN	BT	BOLIVIA	BO
BOTSWANA	BW	BOUVET ISLAND	BV
BRAZIL	BR	BRITISH INDIAN OCEAN TERRITORY	IO
BRUNEI DARUSSALAM	BN	BULGARIA	BG
BURKINA FASO	BF	BURMA	BU
BURUNDI	BI	BYELORUS	BY
CAMEROON	CM	CANADA	CA
CAPE VERDE	CV	CAYMAN ISLANDS	KY
CENTRAL AFRICAN REPUBLIC	CF	CHAD	TD
CHILE	CL	CHINA	CN
CHRISTMAS ISLAND	CX	COCOS (KEELING) ISLANDS	CC
COLOMBIA	CO	COMOROS	KM
CONGO	CG	COOK ISLANDS	CK
COSTA RICA	CR	COTE D'IVOIRE	CI
CUBA	CU	CYPRUS	CY
CZECHOSLOVAKIA	CS	DENMARK	DK
DJIBOUTI	DJ	DOMINICA	DM
DOMINICAN REPUBLIC	DO	EAST TIMOR[1]	TP

[1]Provisional name.

Country Name	Code	Country Name	Code
ECUADOR	EC	EGYPT	EG
EL SALVADOR	SV	EQUATORIAL GUINEA	GQ
ETHIOPIA	ET	FALKLAND ISLANDS (MALVINAS)	FK
FAROE ISLANDS	FO	FIJI	FJ
FINLAND	FI	FRANCE	FR
FRENCH GUIANA	GF	FRENCH POLYNESIA	PF
FRENCH SOUTHERN TERRITORIES	TF	GABON	GA
GAMBIA	GM	GERMAN DEMOCRATIC REPUBLIC	DD
GERMANY, FEDERAL REPUBLIC OF	DE	GHANA	GH
GIBRALTAR	GI	GREECE	GR
GREENLAND	GL	GRENADA	GD
GUADELOUPE	GP	GUAM	GU
GUATEMALA	GT	GUINEA	GN
GUINEA-BISSAU	GW	GUYANA	GY
HAITI	HT	HEARD AND MCDONALD ISLANDS	HM
HONDURAS	HN	HONG KONG	HK
HUNGARY	HU	ICELAND	IS
INDIA	IN	INDONESIA	ID
IRAN (ISLAMIC REPUBLIC OF)	IR	IRAQ	IQ
IRELAND	IE	ISRAEL	IL
ITALY	IT	JAMAICA	JM
JAPAN	JP	JORDAN	JO
KAMPUCHEA, DEMOCRATIC	KH	KENYA	KE
KIRIBATI	KI	KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP
KOREA, REPUBLIC OF	KR	KUWAIT	KW
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LA	LEBANON	LB

Country Name	Code	Country Name	Code
LESOTHO	LS	LIBERIA	LR
LIBYAN ARAB JAMAHIRIYA	LY	LIECHTENSTEIN	LI
LUXEMBOURG	LU	MACAU	MO
MADAGASCAR	MG	MALAWI	MW
MALAYSIA	MY	MALDIVES	MV
MALI	ML	MALTA	MT
MARSHALL ISLANDS	MH	MARTINIQUE	MQ
MAURITANIA	MR	MAURITIUS	MU
MEXICO	MX	MICRONESIA	FM
MONACO	MC	MONGOLIA	MN
MONTSERRAT	MS	MOROCCO	MA
MOZAMBIQUE	MZ	NAMIBIA	NA
NAURU	NR	NEPAL	NP
NETHERLANDS	NL	NETHERLANDS ANTILLES	AN
NEUTRAL ZONE	NT	NEW CALEDONIA	NC
NEW ZEALAND	NZ	NICARAGUA	NI
NIGER	NE	NIGERIA	NG
NIUE	NU	NORFOLK ISLAND	NF
NORTHERN MARIANA ISLANDS	MP	NORWAY	NO
OMAN	OM	PAKISTAN	PK
PALAU	PW	PANAMA	PA
PAPUA NEW GUINEA	PG	PARAGUAY	PY
PERU	PE	PHILIPPINES	PH
PITCAIRN	PN	POLAND	PL
PORTUGAL	PT	PUERTO RICO	PR
QATAR	QA	REUNION	RE
ROMANIA	RO	RWANDA	RW
ST. HELENA	SH	SAINT KITTS AND NEVIS	KN
SAINT LUCIA	LC	ST. PIERRE AND MIQUELON	PM
SAINT VINCENT AND THE GRENADINES	VC	SAMOA	WS
SAN MARINO	SM	SAO TOME AND PRINCIPE	ST

Country Name	Code	Country Name	Code
SAUDI ARABIA	SA	SENEGAL	SN
SEYCHELLES	SC	SIERRA LEONE	SL
SINGAPORE	SG	SOLOMON ISLANDS	SB
SOMALIA	SO	SOUTH AFRICA	ZA
SPAIN	ES	SRI LANKA	LK
SUDAN	SD	SURINAME	SR
SVALBARD AND JAN MAYEN ISLANDS	SJ		
SWAZILAND	SZ	SWEDEN	SE
SWITZERLAND	CH	SYRIAN ARAB REPUBLIC	SY
TAIWAN, PROVINCE OF CHINA	TW	TANZANIA, UNITED REPUBLIC OF	TZ
THAILAND	TH	TOGO	TG
TOKELAU	TK	TONGA	TO
TRINIDAD AND TOBAGO	TT	TUNISIA	TN
TURKEY	TR	TURKS AND CAICOS ISLANDS	TC
TUVALU	TV	UGANDA	UG
UKRAINE	UA	UNITED ARAB EMIRATES	AE
UNITED KINGDOM	GB	UNITED STATES	US
UNITED STATES MINOR OUTLYING ISLANDS	UM	URUGUAY	UY
RUSSIA	SU	VANUATU	VU
VATICAN CITY STATE	VA	VENEZUELA	VE
VIET NAM	VN	VIRGIN ISLANDS (BRITISH)	VG
VIRGIN ISLANDS (U.S.)	VI	WALLIS AND FUTUNA ISLANDS	WF
WESTERN SAHARA[1]	EH	YEMEN	YE
YEMEN, DEMOCRATIC	YD	YUGOSLAVIA	YU
ZAIRE	ZR	ZAMBIA	ZM
ZIMBABWE	ZW		

[1]Provisional name.

Table E-2 shows the codes that are part of a reserved code list. These codes were deleted from ISO 3166 in 1981.

Table E–2. Deletions from ISO 3166 in 1981

Country Name	Code
CANTON AND ENDERBURY ISLANDS	CT
DRONNING MAUD LAND	NQ
JOHNSTON ISLAND	JT
MIDWAY ISLANDS	MI
WAKE ISLAND	WK
PACIFIC ISLANDS	PC
UNITED STATES MISCELLANEOUS PACIFIC ISLANDS	PU

E.2 T.61 Syntax

Table E-3 shows the set of valid T.61 characters. (The row headings indicate the lower four bits, and the column headings show the higher four bits of the encoding in hexadecimal.)

Note: The 1) entry in the table indicates that it is not recommended that you use the codes in Column 2, Row 3, and Column 2, Row 4. Instead, use the appropriate code in Column A.

Table E–3. T.61 Syntax

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P		p				°			Ω	K
1			!	1	A	Q	a	q			i	±	`		Æ	æ
2			"	2	B	R	b	r			¢	2	´		Đ	đ
3			1)	3	C	S	c	s			£	3	^		a	ð
4			1)	4	D	T	d	t			$	x	~		Ħ	ħ
5			%	5	E	U	e	u			¥	µ	¯			ı
6			&	6	F	V	f	v			#	¶	˘		Ĳ	ij
7			'	7	G	W	g	w			§	•	¸		L·	l·
8			(	8	H	X	h	x			¤	+	¨		Ł	ł
9			)	9	I	Y	i	y							Ø	ø
A			*	:	J	Z	j	z					°		Œ	œ
B			+	;	K	[	k				<<	>>	˛		º	ß
C			,	<	L		l					¼	_		Þ	þ
D			–	=	M	]	m					½	"		Ŧ	ŧ
E			·	>	N		n					¾	˛		ŋ	ŋ
F			/	?	O	_	o					¿	ˇ		'n	

The administration interface supports only characters smaller than 0x7e for names. The X/Open Directory Service (XDS) Application Programming Interface (API) supports the full T.61 range as indicated in Table E-3.

Some T.61 alphabetical characters have a two-byte representation. For example, a lowercase letter "a" with acute accent is represented by 0xc2 (code for acute accent) followed by 0x61 (code for lowercase "a").

Only certain combinations of diacritical characters and basic letters are valid. They are shown in Table E-4.

Table E–4. Combinations of Diacritical Characters and Basic Letters

Name	Repr.	Code	Valid Basic Letters Following
grave accent	`	0xc1	a, A, e, E, i, I, o, O, u, U
acute accent	´	0xc2	a,A,c,C,e,E,g,i,I,l,L,n,N,o,O,r,R, s,S,u,U,y,Y,z,Z
circumflex accent	^	0xc3	a,A,c,C,e,E,g,G,h,H,i,I,j,J,o,O,s,S, u,U,w,W,y,Y
tilde	~	0xc4	a,A,i,I,n,N,o,O,u,U
macron	¯	0xc5	a,A,e,E,i,I,o,O,u,U
breve	˘	0xc6	a, A, g, G, u, U
dot above	˙	0xc7	c,C,e,E,g,G,I,z,Z
umlaut	¨	0xc8	a,A,e,E,i,I,o,O,u,U,y,Y
ring	°	0xca	a, A, u, U
cedilla	¸	0xcb	c,C,G,k,K,l,L,n,N,r,R,s,S,t,T
double accent	"	0xcd	o, O, u, U
ogonek	˛	0xce	a, A, e, E, i, I, u, U
caron	ˇ	0xcf	c,C,d,D,e,E,l,L,n,N,r,R,s,S,t,T,z,Z

The nonspacing underline (code 0xcc) must be followed by a Latin alphabetical character, that is, a basic letter (a to z or A to Z), or a valid diacritical combination.

Appendix F

Worksheets

This appendix contains seven sample GDS configuration worksheets that administrators can copy for their own use. Admininstrators are not obliged to use every worksheet; they are provided to help organize the information required to initialize and configure GDS and to make modifications at a later date.

ACL Object Entry Worksheet

Directory Entry: _____

Access Class	**DN of User**	**Interpretation** (Single Object or Subtree)
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Directory Entry: _____

Access Class	**DN of User**	**Interpretation** (Single Object or Subtree)
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

ACL Schema Worksheet

Attribute Type	Access Class		
	PUBLIC	STANDARD	SENSITIVE

Cell Worksheet

Global Cell Name: _____

CDS-Cell Attribute

Namespace UUID _____
Root dir name _____
Root dir UUID _____

Cell Replica Attributes

Replica Type _____
Clearinghouse UUID _____
Clearinghouse name _____
Tower 1 _____
Tower 2 _____
Tower 3 _____
Tower 4 _____
Tower 5 _____

CDS-Cell Attribute

Namespace UUID _____
Root dir name _____
Root dir UUID _____

Cell Replica Attributes

Replica Type _____
Clearinghouse UUID _____
Clearinghouse name _____
Tower 1 _____
Tower 2 _____
Tower 3 _____
Tower 4 _____
Tower 5 _____

Client Worksheet

Global Cell Name:_____

Name of Client Machine:_____

PSAP Address (Client Stub):

NSAP Address _____

Transport Procotol _____

T-Selector _____ P-Selector _____ S-Selector _____

DUA Cache Information

DN of DSA	DSA-Type	General DSA Type (Remote GDS, Remote Non-GDS, Initial, First-level)

Global Cell Name:_____

Name of Client Machine:_____

PSAP Address (client stub):

NSAP Address _____

Transport Procotol _____

T-Selector _____ P-Selector _____ S-Selector _____

DUA Cache Information

DN of DSA	DSA-Type	General DSA Type (Remote GDS, Remote Non-GDS, Initial, First-level)

Client/Server Worksheet

Global Cell Name:_____

Name of Client/Server Machine:_____

DN of DSA: _____

PSAP Address:

NSAP Address _____

Transport Procotol _____

T-Selector _____ P-Selector _____ S-Selector _____

DUA Cache Information

DName of DSA	DSA-Type	General DSA Type (Remote GDS, Remote Non-GDS, Initial, First-level)

Global Cell Name:_____

Name of Client/Server Machine:_____

DN of DSA: _____

PSAP Address:

NSAP Address _____

Transport Procotol _____

T-Selector _____ P-Selector _____ S-Selector _____

DUA Cache Information

DN of DSA	DSA-Type	General DSA Type (Remote GDS, Remote Non-GDS, Initial, First-level)

GDS Remote and Non-GDS DSA Worksheet

DN of DSA: ______________________ DSA Type ____________

PSAP Address (DSA)

NSAP Address ______________________

Transport Protocol ______________________

T-Selector ____________ P-Selector ____________ S-Selector ____________

DN of DSA: ______________________ DSA Type ____________

PSAP Address (DSA)

NSAP Address ______________________

Transport Protocol ______________________

T-Selector ____________ P-Selector ____________ S-Selector ____________

DN of DSA: ______________________ DSA Type ____________

PSAP Address (DSA)

NSAP Address ______________________

Transport Protocol ______________________

T-Selector ____________ P-Selector ____________ S-Selector ____________

DN of DSA: ______________________ DSA Type ____________

PSAP Address (DSA)

NSAP Address ______________________

Transport Protocol ______________________

T-Selector ____________ P-Selector ____________ S-Selector ____________

Shadow Update Worksheet

Initiator: _____

Target (Shadow DSA): _____

DNs of Entries	**Interpretation** (object/subtree)	**Update Frequency** (HIGH, LOW, or MEDIUM)	**Update Time** (minutes, hours, or days)

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Notes

Notes

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 Administration Guide — Extended Services

TITLES IN THE OSF™DCE SERIES:

Introduction to OSF™DCE

OSF™DCE User's Guide and Reference

OSF™DCE Administration Guide
— Introduction
— Core Components
— Extended Services

OSF™DCE Administration Reference

OSF™DCE Application Development Guide

OSF™DCE Application Development Reference

Application Environment Specification (AES)
Distributed Computing

Printed in the U.S.A.

Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142

Prentice Hall, Inc.

DISTRIBUTED SYSTEMS

ISBN 0-13-176561-2

9 780131 765610 90000