DCE 1.2.2 GDS Administration Guide and Reference

OSF® DCE Documentation

The Open Group

OSF® DCE Product Documentation:

DCE 1.2.2 GDS Administration Guide and Reference

ISBN 1–85912–133–0

Document Number F211

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

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

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

or by Electronic Mail to:
OGPubs@opengroup.org

OTHER NOTICES

Contents

List of Figures

List of Tables

Preface

The Open Group

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

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

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

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

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

The Development of Product Standards

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

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

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

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

Open Group Publications

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

There are several types of specification:

CAE Specifications

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

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

Preliminary Specifications

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

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

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

Consortium and Technology Specifications

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

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

In addition, The Open Group publishes:

Product Documentation

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

Guides

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

Technical Studies

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

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

Versions and Issues of Specifications

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

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

Corrigenda

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

Ordering Information

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

This Book

The *DCE 1.2.2 GDS Administration Guide and Reference* provides concepts and procedures that enable you to manage the OSF® Distributed Computing Environment (DCE) Global Directory Service (GDS). Basic DCE terms are introduced throughout the *DCE 1.2.2 GDS Administration Guide and Reference*. A glossary for all of the DCE documentation is provided in the *DCE 1.2.2 Introduction to OSF DCE*. The *DCE 1.2.2 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 document applies to the OSF® DCE Version 1.2.2 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 GDS. After reading the guide, you will understand what the system administrator needs to do to plan for DCE GDS. Once you have built the DCE GDS source code on your system, use this guide to assist you in installing executable files and configuring DCE. The *DCE 1.2.2 Release Notes* contain instructions for installing and building DCE source code.

Related Documents

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

- *DCE 1.2.2 Introduction to OSF DCE*
  Document Number F201, ISBN 1–85912–182–9
- *DCE 1.2.2 Command Reference*
  Document Number F212, ISBN 1–85912–138–1
- *DCE 1.2.2 Application Development Reference*
  Document Number F205A, ISBN 1–85912–103–9 (Volume 1)
  Document Number F205B, ISBN 1–85912–108–X (Volume 2)
  Document Number F205C, ISBN 1–85912–159–4 (Volume 3)
- *DCE 1.2.2 Application Development—Introduction and Style Guide*
  Document Number F202, ISBN 1–85912–187–X
- *DCE 1.2.2 Application Development Guide—Core Components*
  Document Number F203A, ISBN 1–85912–192–6 (Volume 1)
  Document Number F203B, ISBN 1–85912–154–3 (Volume 2)
- *DCE 1.2.2 Application Development Guide—Directory Services*
  Document Number F204, ISBN 1–85912–197–7
- *DCE 1.2.2 DFS Administration Guide and Reference*
  Document Number F209A, ISBN 1–85912–123–3 (Volume 1)
  Document Number F209B, ISBN 1–85912–128–4 (Volume 2)
- The *DCE 1.2.2 Administration Guide—Introduction*
  Document Number F207, ISBN 1–85912–113–6
- The *DCE 1.2.2 Administration Guide—Core Components*
  Document Number F208, ISBN 1–85912–118–7
- *DCE 1.2.2 File-Access Administration Guide and Reference*
  Document Number F216, ISBN 1–85912–158–6
- *DCE 1.2.2 File-Access User's Guide*
  Document Number F217, ISBN 1–85912–163–3
- *DCE 1.2.2 Problem Determination Guide*
  Document Number F213A, ISBN 1–85912–143–8 (Volume 1)
  Document Number F213B, ISBN 1–85912–148–9 (Volume 2)

- *DCE 1.2.2 Testing Guide*
  Document Number F215, ISBN 1–85912–153–5
- *DCE 1.2.2 File-Access FVT User's Guide*
  Document Number F210, ISBN 1–85912–189–6
- *DCE 1.2.2 Release Notes*
  Document Number F218, ISBN 1–85912–168–3

For a detailed description of OSF DCE documentation, see the *DCE 1.2.2 Introduction to OSF DCE* .

Typographic and Keying Conventions

This guide uses the following typographic conventions:

Bold	**Bold** words or characters represent system elements that you must use literally, such as commands, options, and pathnames.
Italic	*Italic* words or characters represent variable values that you must supply. *Italic* type is also used to introduce a new DCE term.
`Constant width`	Examples and information that the system displays appear in `constant width` typeface.
[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
…	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

This guide uses the following keying conventions:

<Ctrl-*x* **>** or ^*x*

The notation **<Ctrl-** *x* **>** or ^*x* followed by the name of a key indicates a control character sequence. For example, **<Ctrl-C** means that you hold down the control key while pressing **<C>**.

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

Problem Reporting

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

Pathnames of Directories and Files in DCE Documentation

For a list of the pathnames for directories and files referred to in this guide, see the *DCE 1.2.2 Administration Guide—Introduction* and *DCE 1.2.2 Testing Guide*.

Trademarks

Motif®, OSF/1®, and UNIX® are registered trademarks and the IT DialTone™, The Open Group™, and the "X Device"™ are trademarks of The Open Group.

DEC, DIGITAL, and ULTRIX are registered trademarks of Digital Equipment Corporation.

DECstation 3100 and DECnet are trademarks of Digital Equipment Corporation.

HP, Hewlett-Packard, and LaserJet are trademarks of Hewlett-Packard Company.

Network Computing System and PasswdEtc are registered trademarks of Hewlett-Packard Company.

AFS, Episode, and Transarc are registered trademarks of the Transarc Corporation.

DFS is a trademark of the Transarc Corporation.

Episode is a registered trademark of the Transarc Corporation.

Ethernet is a registered trademark of Xerox Corporation.

AIX and RISC System/6000 are registered trademarks of International Business Machines Corporation.

IBM is a registered trademark of International Business Machines Corporation.

DIR-X is a trademark of Siemens Nixdorf Informationssysteme AG.

MX300i is a trademark of Siemens Nixdorf Informationssysteme AG.

NFS, Network File System, SunOS and Sun Microsystems are trademarks of Sun Microsystems, Inc.

PostScript is a trademark of Adobe Systems Incorporated.

Microsoft, MS-DOS, and Windows are registered trademarks of Microsoft Corp.

NetWare is a registered trademark of Novell, Inc.

Part 1

OSF DCE GDS Administration Guide

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

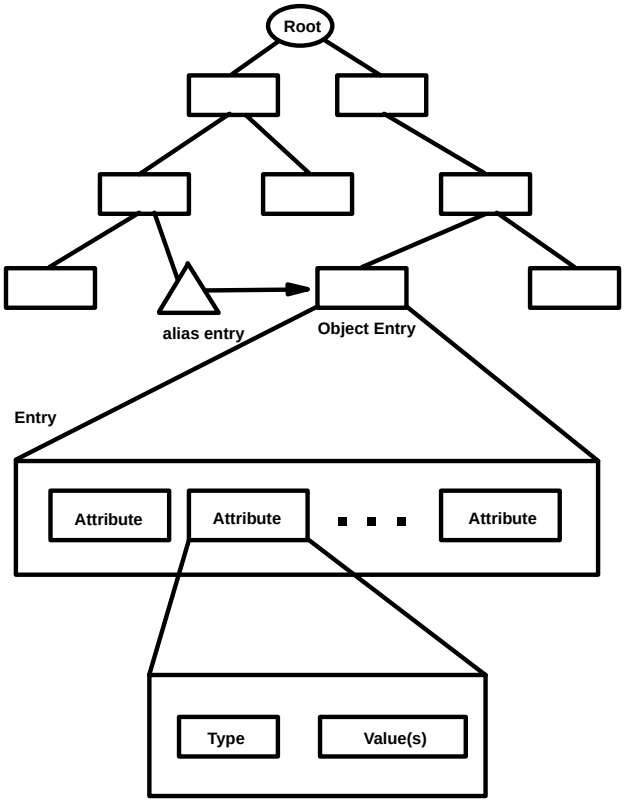
information about that object in the DIB. An *alias* entry is a special type of entry that provides an alternative name for an entry. Alias entries are described later in this chapter.

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 their last name, full name, postal address, telephone number, 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. The attribute types for an entry describing a person, 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. Structure of the DIB



Each attribute type also has an *attribute syntax* which describes the format of the legal values of that attribute type. An example is *Common-Name*, which is assigned the **Case Ignore String** syntax. This allows a **Printable String** or a **Teletex String** to be entered and specifies that cases are not significant for matching. **Printable String** or **Teletex String** defines the permitted character sets.

1.1.3 Object Classes

In order to categorize entries in the DIB, each entry is said to belong to one or more *object classes*. 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.)

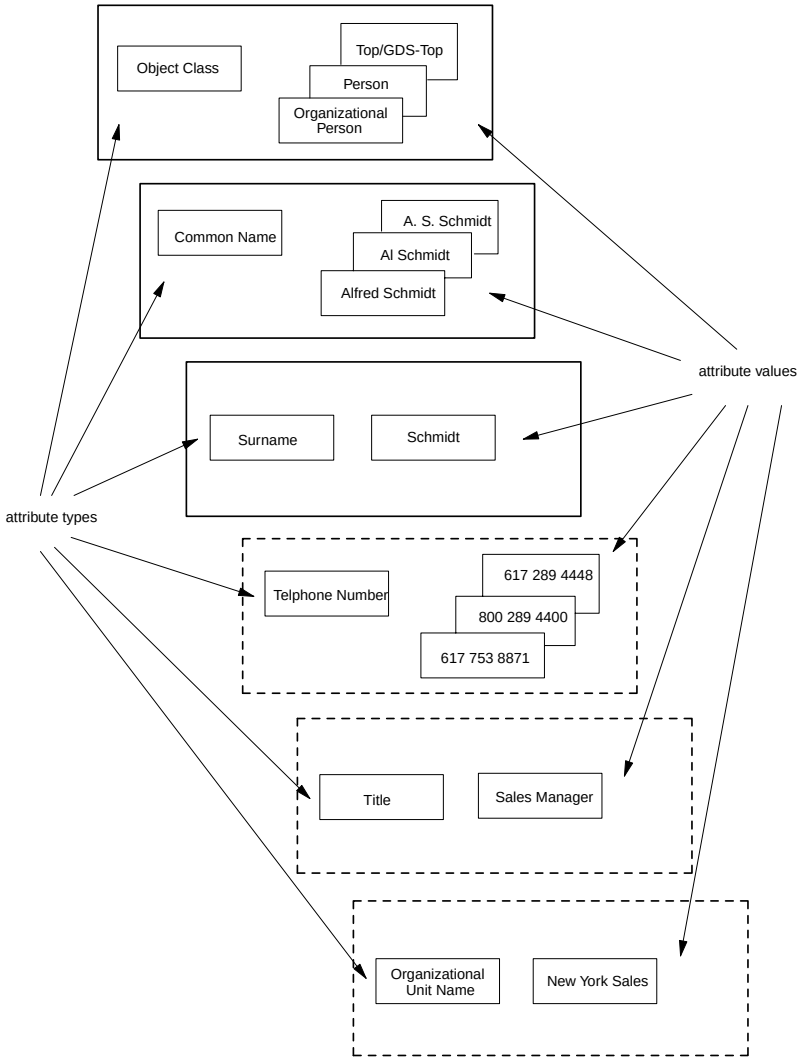
For example, an entry representing an organization must contain an attribute called **Organization-Name**, which has the name of the organization as its value. It can contain optional attributes that describe the organization, the state or locality in which the organization resides, the postal address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



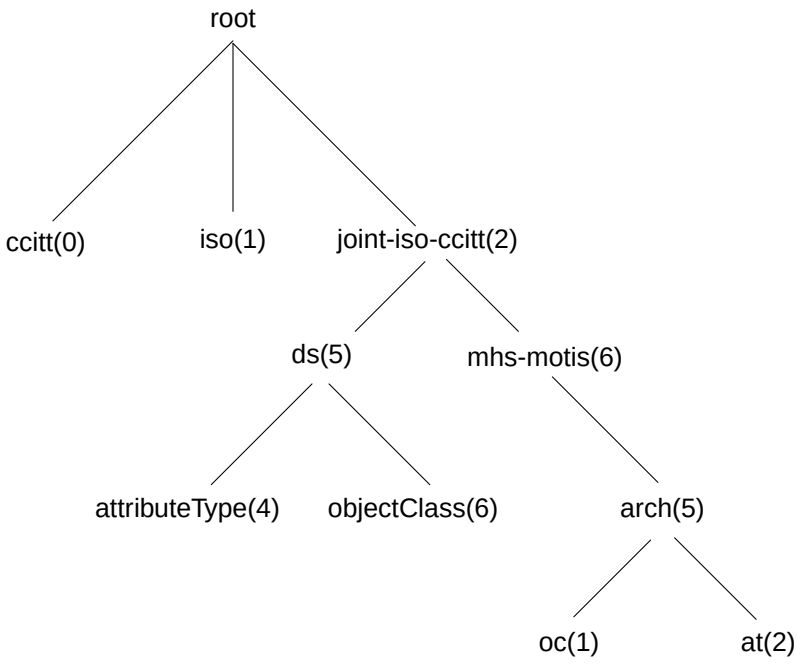
1.1.4 Object Identifiers

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.

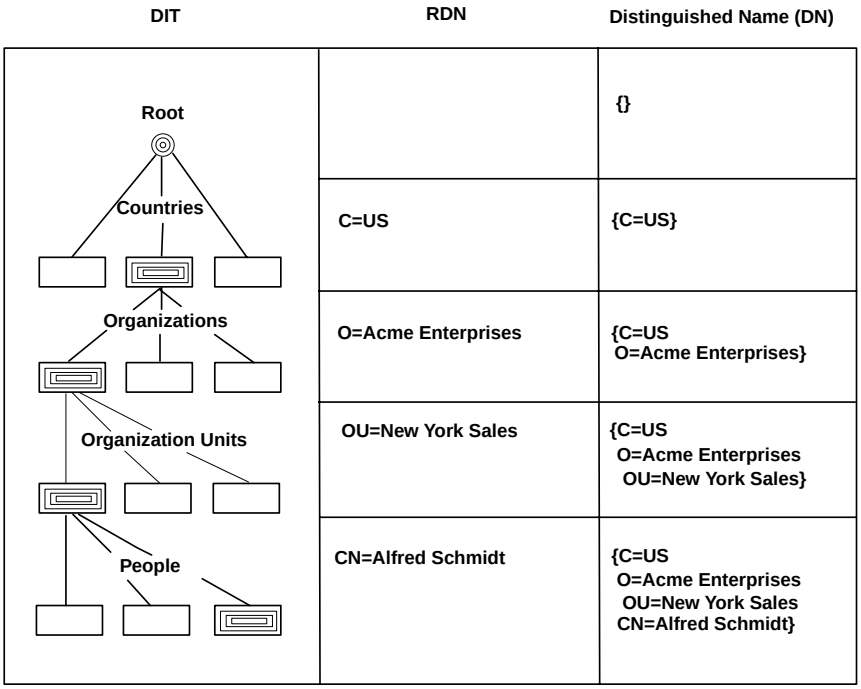
1.2.1 Distinguished Names

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. 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

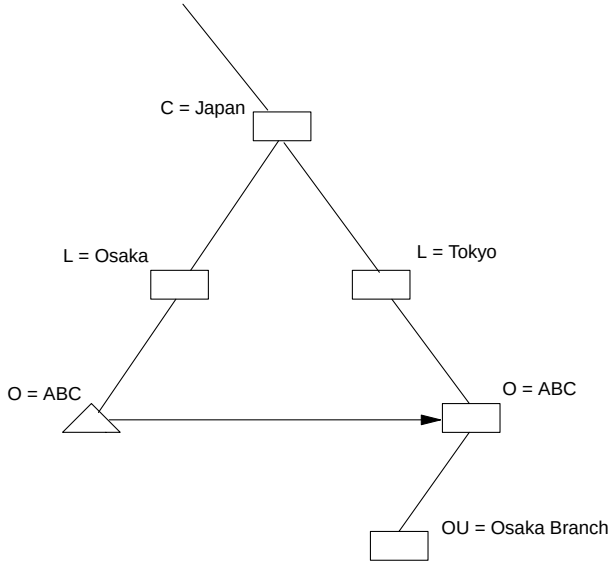
Alternative names or aliases are supported in the DIT through special pointer entries called *alias entries*. Alias entries do not contain any attributes other than those attributes belonging to their RDN, the object class attribute, and the aliased object name attribute (that is, the DN of the aliased 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.

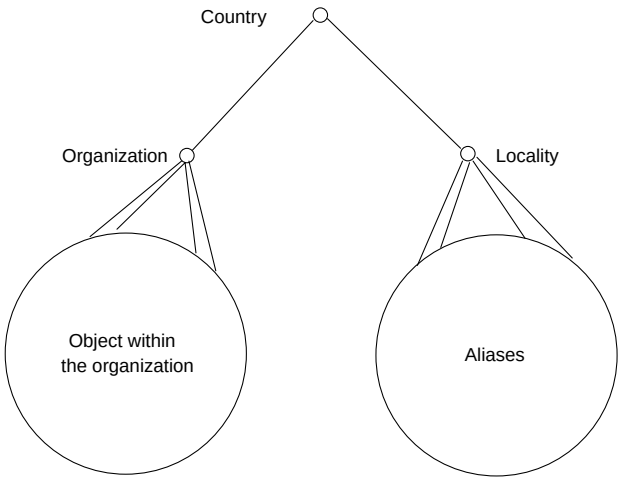
To make the transition easier for directory users and to guarantee that a search based on the old information finds its target, an alias for **O=ABC** is added to the directory beneath **L=Osaka**. This alias entry points to the object entry **O=ABC**. A search for **ABC** under **L=Osaka** in the DIT 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. 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 Standardized Operations and Features

The following table 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. (Refer to the *DCE 1.2.2 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

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:

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

1.3.1 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*.

1.3.2 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.

1.3.3 Searching

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

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.

1.3.4 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).

1.3.5 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.

1.3.6 User Identification (Authentication)

The directory service supports simple user authentication using name and password. GDS administration programs do not use strong authentication. However, XDS application programs may use the Strong Authentication Package as a security method. Refer to the *DCE 1.2.2 Application Development Guide* for more information on how the SAP can be integrated into an XDS application program.

1.3.7 Transparent Routing of Requests

The directory service supports distribution transparency in which the user gets the same consistent view of information irrespective of where the user is attached to the directory service. 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.

Refer to section 1.6 for a detailed description of referral and chaining.

1.4 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 Replication of Information and shadow update
- Access Control
- DCE Authentication
- Directory User Agent Cache
- Remote Administration
- Tree Processing
- Configurable Routing of Requests

1.4.1 Shadow Information

Frequently used data that is stored on one DSA can be copied and updated to 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.

1.4.2 Directory Distribution and Knowledge Information

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 must have 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 1-7, 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.

GDS models the knowledge information so that each DSA contains an entry with its own name (**/C=DE/O=Smith Ltd/OU=DI/CN=DSA/CN=DSA1**, for example) 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 1-7 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**)

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 1-7, 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 1-7 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.

Mandatory shadow entries will be modified automatically if their corresponding master 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 1-7 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 1-7 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 1-9 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** .

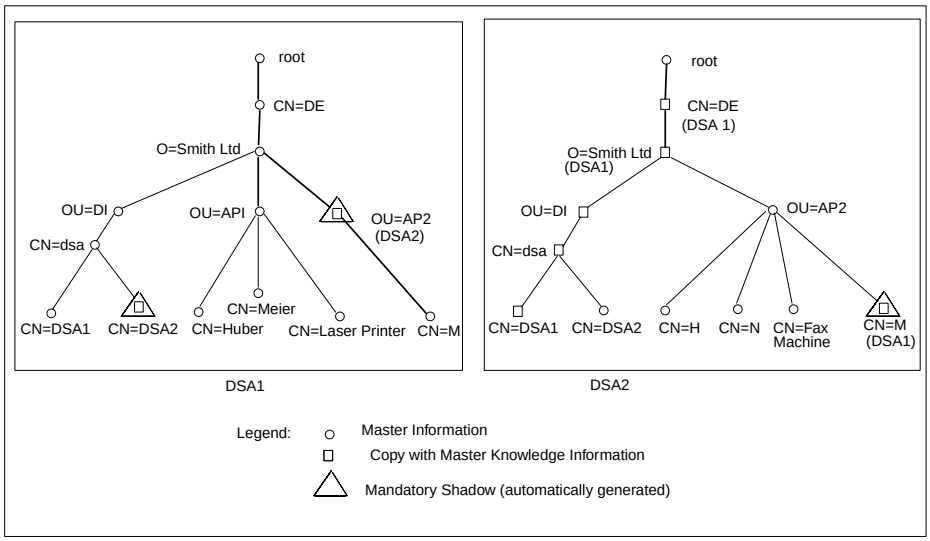
- The shadows of the DSA entries that appear as **Master Knowledge** attributes in the shadows available in this DSA.

  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 1-7, **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 does not restrict the request to **LOCAL SCOPE**, **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 1.4.1.)

Figure 1–7. Storage of Knowledge Information in GDS



1.4.3 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.

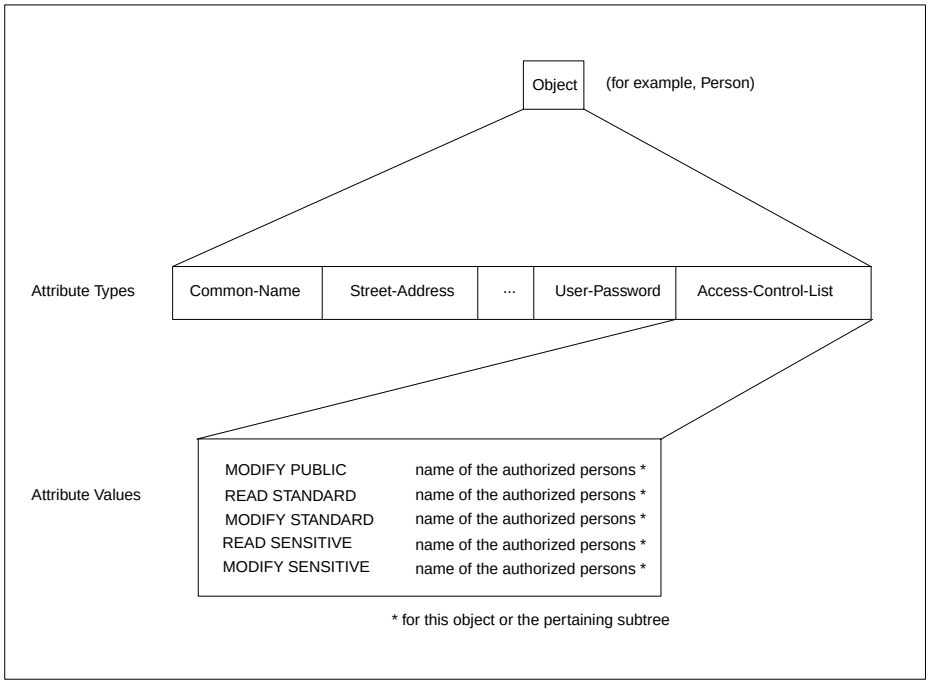
Because there can be many different attributes in the DIT, it is too expensive to define a protection mechanism for each individual attribute type. Instead, the attributes are divided into three classes:

- 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 a single value. This value is composed of 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.

The following figure 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 1–8. Access Control using the ACL Attribute



The following specifications apply for assigning access rights:

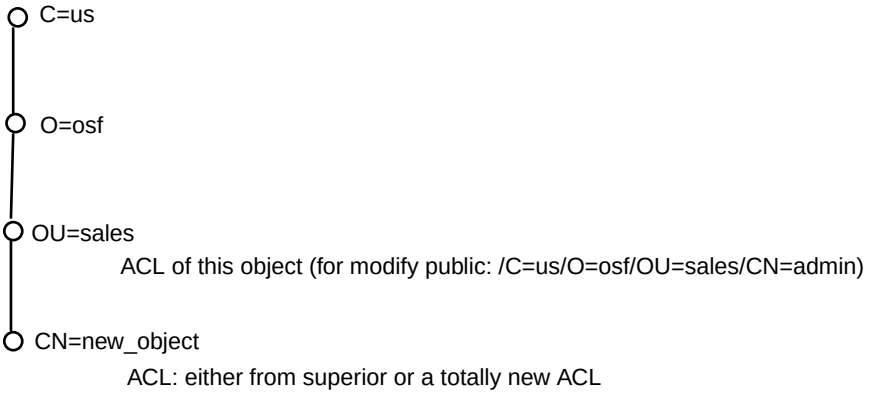
- The ACL of the default schema does not restrict access rights when GDS is configured. Every user, including the anonymous user (one without a DN and password), has read and write access to all attributes in the schema. When an object is created without supplying the ACL attribute, the ACL attribute is inherited from the superior object. For first-level objects (directly under root) the ACL attribute is inherited from the schema object.

  The administrator needs to remember that any anonymous user who has access to the GDS administration programs can log on to a remote DSA. However, all 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.

  In Figure 1-9, **CN=new_object** can only be created if the administrator logs on 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 1–9. 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 1-9, **CN=new_object** can only be deleted if the administrator logs on as **/C=us/O=osf/OU=sales/CN=admin**.

  The object can only be modified if the administrator logs on 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.

1.4.3.1 Trusted DSA Tables

The administrator can restrict authenticated access by specific DSAs by creating and maintaining the Trusted DSA Table (TDT). These restrictions apply if a chained operation follows a bind initiated by the intermediate DSA using the DCE authentication mechanism. Each table entry contains the DN of a DSA and permission tags that define the type of access rights permitted for the DSA.

Refer to section 3.7 in chapter 3, *Developing a Configuration Plan* in this manual.

1.4.4 DCE Authentication

In addition to simple authentication, GDS supports DCE Authentication as a security method for accessing DSAs in the DCE environment. DCE authentication requires that users and the DSAs these users want to access are registered in the DCE registry. The extended attributes feature of the DCE Registry is used for this purpose.

The DCE rgy schema contains two GDS specific attributes (by default):

X500_DN Present for principals representing GDS users and DSAs. This attribute has to contain the DN of the principal for using the Directory Service. The DN must be entered in Printable Form syntax (**/C=de/O=sni/ OU=buba, OU=nm123/CN=meier**, for example).

X500_DSA_Admin
Only necessary for the administrators principals. This attribute has to contain the list of the names of all DSAs on which the administrator expects to create shadows.

Users will have to logon to the DCE (for example, using **dce_login**)prior to using the DCE authentication in a GDS client program.

1.4.5 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 on to the local machine using the option in Mask 1 **Logon to the DUA cache**. (Masks enable a user to enter input interactively in the GDS administration programs. Masks are described in detail in Chapters 8-11.)

1.4.6 Remote Administration

The X.500 standard does not cover all aspects of Directory administration. This means that each DSA must be administered separately. For maximum flexibility, GDS provides remote administration. This enables administrators to log on to other DSAs and perform directory operations from one workstation. GDS enables remote administration of objects, schema, copies, and subtrees.

The administrator remotely logs on to a DSA through the User Identification mask (Mask 1). The administrator enters a DN that represents the administrator and a password. 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.

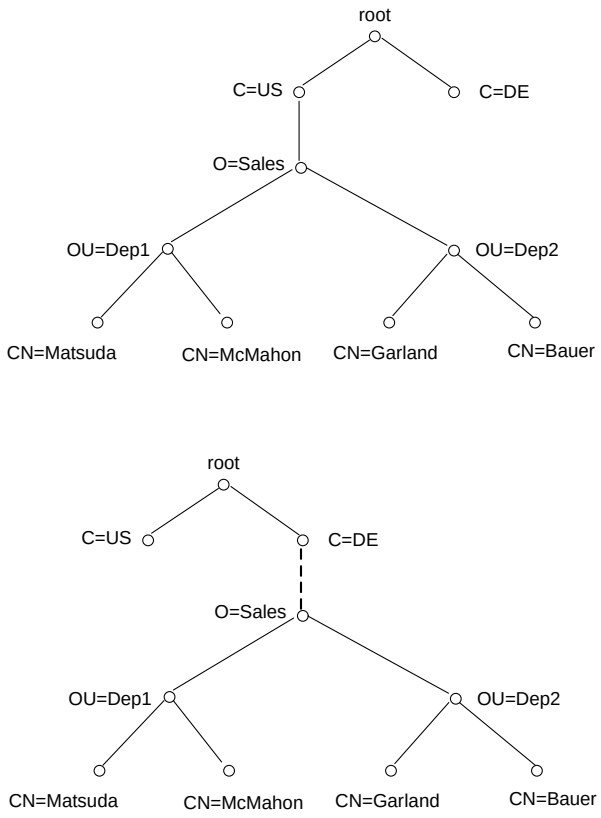
1.4.7 Tree Processing

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 1-10 illustrates how tree processing functions could be used to to delete a subtree from one node and append it to another.

Figure 1–10. Moving a Subtree



1.4.8 Configurable Routing

Configurable routing is an optional feature that does not completely conform to the distributed operations model described later in section 1.6. Configurable routing provides a higher level of control of distributed operations (for example, an administrator can configure a DSA where chaining is not used). Configurable routing is an advanced feature of GDS that is generally only used in large, complex configurations of the Directory Service.

Refer to Appendix J, *The DSA Configuration File* , for detailed information on how to configure routing.

1.5 Schema

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.

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

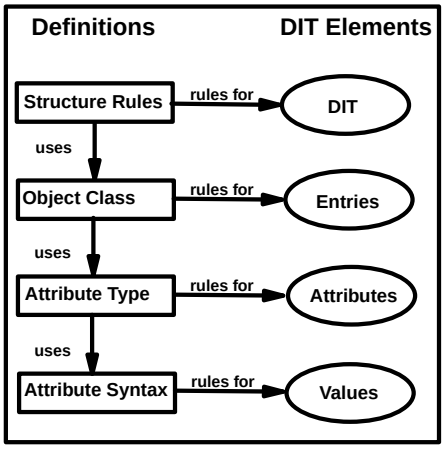
These rules are represented by attribute values of the schema.

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.

Figure 1-11 shows the relationship between schemas and the directory information model.

Figure 1–11. Relationship Between Schemas and the DIT



1.5.1 The GDS Standard Schema

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

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)

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.

1.5.2 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	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	7	CN	DNA
16	2	L	LOC
17	15	CN	REP
18	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. **Organizational Unit** (**OU**) is located beneath **Organization** because it has a Superior Rule Number of 3, and so on.

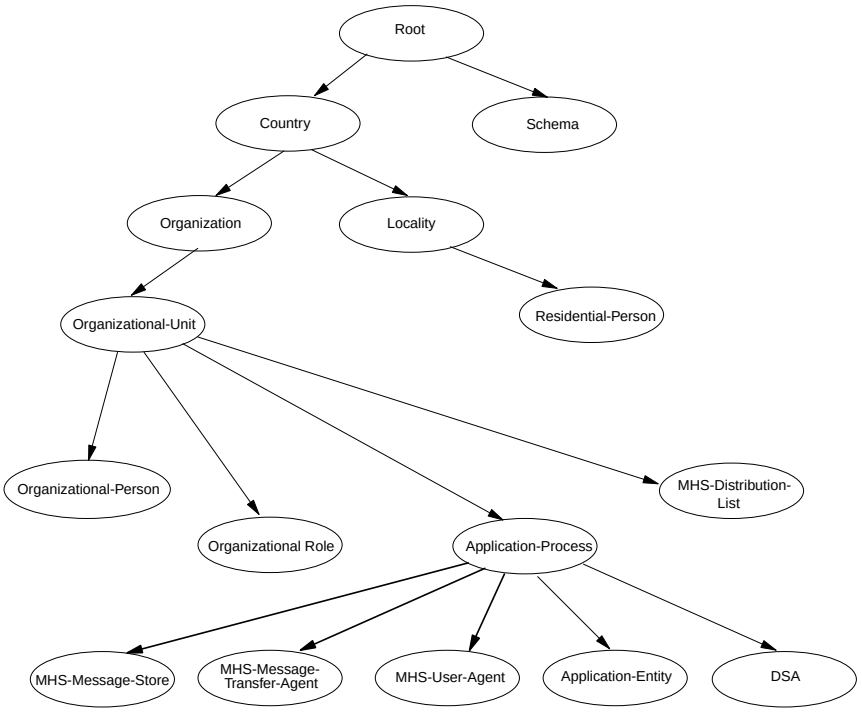
The SRT only contains structural object classes (that is, classes forming branches in the DIT). Other object classes (such as abstract and alias classes) are not included.

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 1-12 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 1-12, the SRT contains two entries for **Organizational-Person** that specify different sets of permitted naming attributes.

Figure 1–12. Structure of the DIT for GDS Administration Programs



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

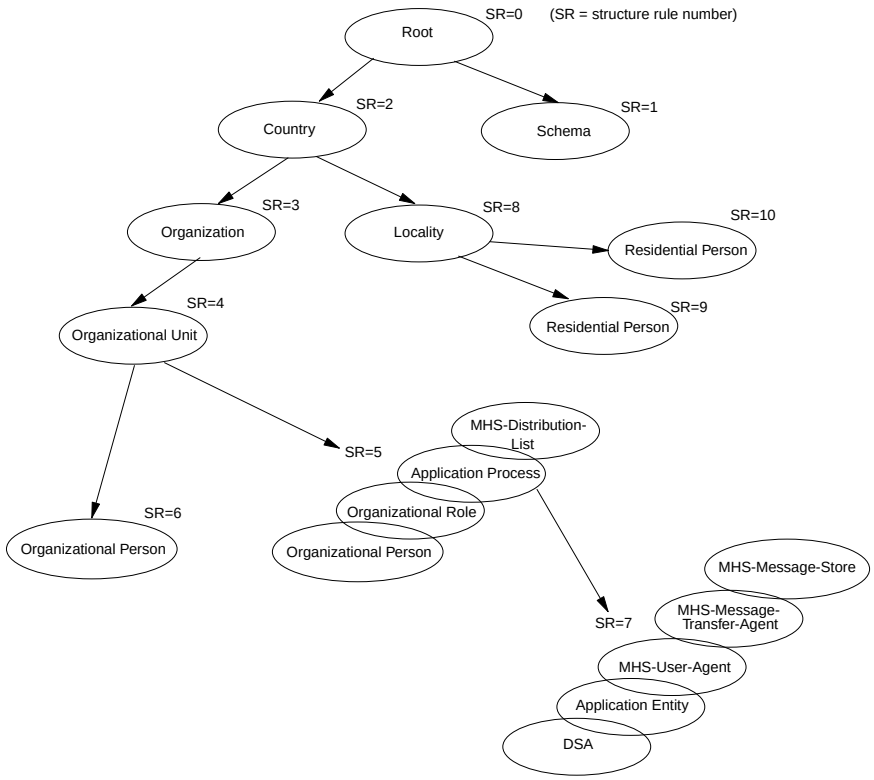
Table 1–3. SRT Entries (for GDS Administration Programs)

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

Rule Number	Superior Rule Number	Acronyms of Naming Attribute	Acronym of Structural Object Class
5	4	CN	ORP, APP, ORR, MDL
6	4	CN, OU	ORP
7	5	CN	DSA, APE, MMS, MTA, MUA, DNA
8	2	L	LOC
9	8	CN	REP
10	8	CN, STA	REP

Figure 1-13 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 1-13, the SRT contains two entries for **Organizational-Person** and **Residential-Person** that specify different sets of permitted naming attributes.

Figure 1–13. Structure of the DIT for Administration Programs

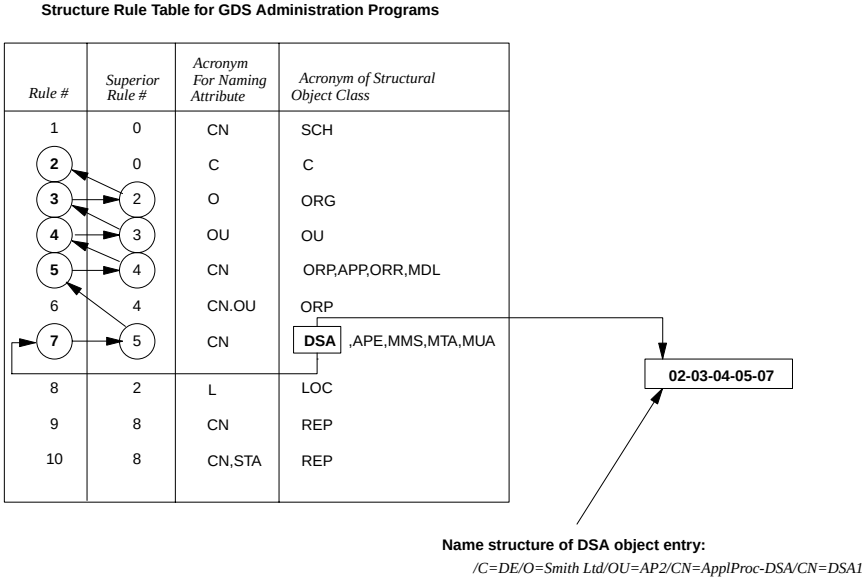


An administrator uses the structure rules to determine the name structure of an object entry so that the DN of the entry can be entered in administration program masks. For example, in Figure 1-14, the name structure of the following DSA object entry is 02-03-04-05-07:

/C=DE/O=Smith Ltd/OU=AP2/CN=ApplProc-DSA/CN=DSA1

Starting from Structure Rule 7 (the Structure Rule for DSA), it is easier to work backwards 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 1–14. Determining the Name Structure of a DSA Object Entry



1.5.3 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 (refer to 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

Object Class				Super-class	OID	File No.	Mandatory Attributes	Optional Attributes
Acronym	**Name**	**Kind**	**Aux.**					
TOP	Top	Abstract	—	None	85.6.0	–1	OCL	None
GTP	GDS-Top	Abstract	—	TOP	None	–1	None	ACL MK
SCH	Schema	Structural	—	GTP	43.12.2.\ 1107.1.3.6.0	0	CN	TST SRT OCT AT
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

Note: All these object identifiers 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 1-12.

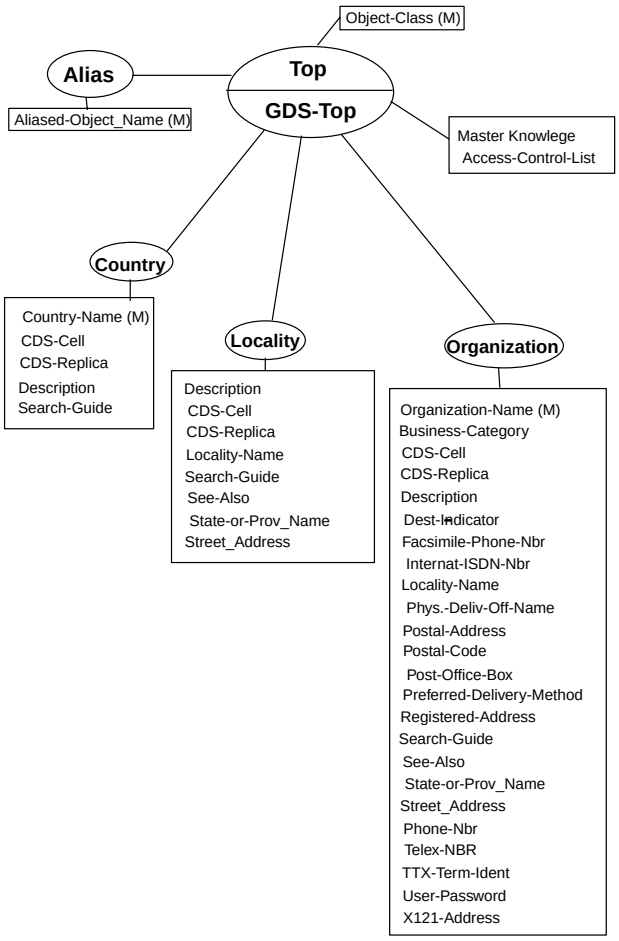
The object class, **Top**, is the root of the tree with **Alias** and **GDS-Top** as the main branches. **Top** contains the attribute type **Object-Class**, which is inherited by all the other object classes.

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 1-12 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 1-15 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 parenthesis indicates a mandatory attribute.

Figure 1–15. 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 Alias	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 these object identifiers 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**.

1.5.3.1 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 1.6.4.2, 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 1.6.4.2), the object classes should not share the file number in order to save space on disk.

1.5.3.2 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 using 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.

1.5.4 The Attribute Table

The Attribute Table (AT) defines the attributes that constitute the entries in the GDS standard schema. (Refer to the *DCE 1.2.2 Administration Guide—Core Components* for a complete listing of the AT.) The OIDs range from 85.4.0 to 85.4.40, 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

Attribute		**OID**	**Lower Bound**	**Upper Bound**	**Max. No. of Val.**	**Syntax**	**Phon. Flag**	**Access Class**	**Index Level**
Acr.	**Name**								
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

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	**Name**								
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

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 constants 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 using the schema administration operations described in Chapter 9.

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

1.5.5 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. (Refer to the *DCE 1.2.2 Application Development Guide* for more information about syntaxes).

Refer to Chapter 9 for a description of the syntaxes that can be applied.

1.6 GDS as a Distributed Service

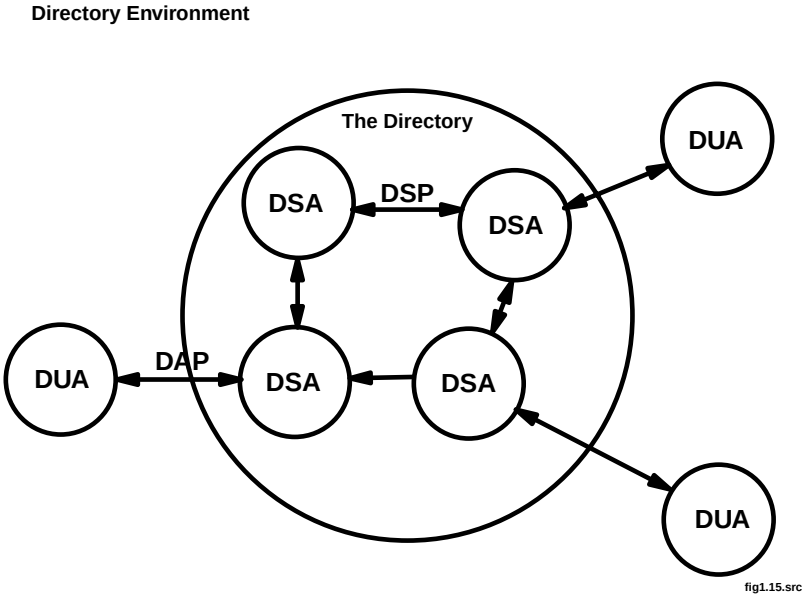
GDS has two basic functions in a DCE cell.

- 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 1-16) cooperate with each other to perform directory service operations. This cooperation often involves the navigation of operations through the network.

Figure 1–16. DSA/DUA Relationship



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 offers a convenient way for a client to download distributed operations to a server which cooperates with other DSAs to perform user requests.

Through DAP and DSP the Directory Service provides transparent service. Users get the same result from a request irrespective of where in the Directory system the request originates.

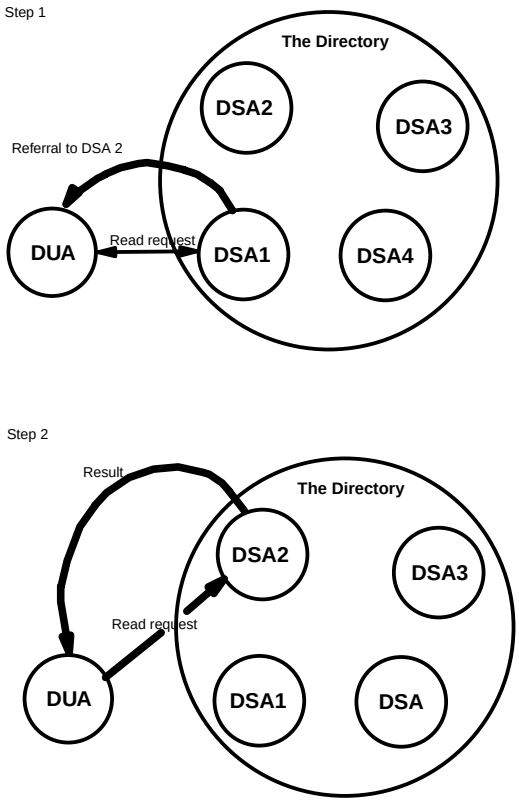
1.6.1 Referral

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 (DAP referral) or the calling DSA (DSP referral) 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 1-17 shows an example of referral. **DSA1** passes a referral to **DSA2** back to **DUA**. **DUA** then makes a request to **DSA2** .

Figure 1–17. Referral



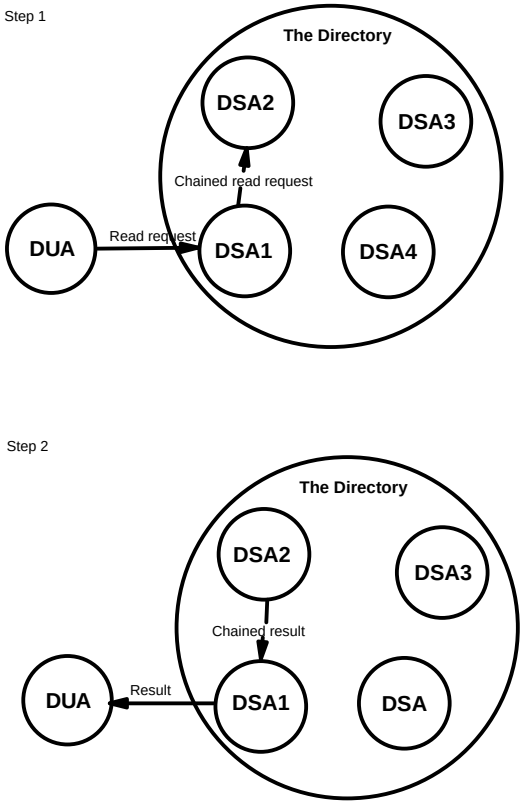
1.6.2 Chaining

As an alternative to the referral mechanism, 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.

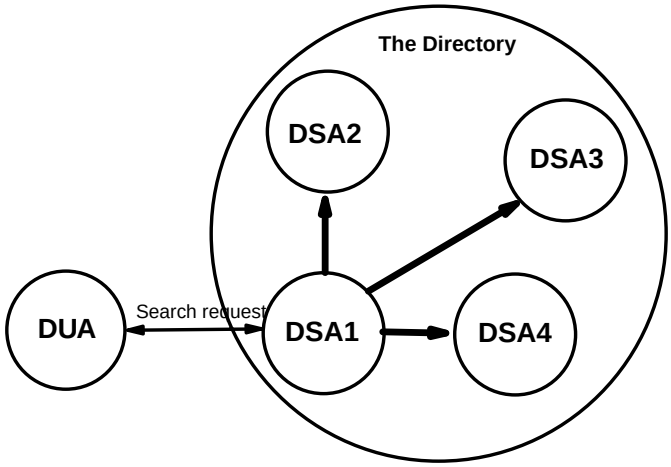
Figure 1-18 shows an example of chaining. **DUA** issues a request to **DSA1**. **DSA1** is unable to service the request and passes it to **DSA2**. **DSA2** services the request, passes the result back to **DSA1**, and **DSA1** passes the result back to **DUA**.

Figure 1–18. Chaining



In the evaluation of a search or list operation a DSA may have to decompose the request in a number of different subrequests and chain them out to other DSAs. This is called request decomposition. Figure 1-19 shows an example of request decomposition. **DSA1** sends a request out in parallel and merges the result before returning it to the DSA.

Figure 1–19. Request Decomposition



1.6.3 Navigation in the GDS

When a service request is generated by a Directory Service user, the bind DSA (the DSA to which the bind was made) attempts to satisfy the request. When a request cannot be satisfied by the bind DSA alone, either the bind DSA or the DUA (acting on information returned by the bind DSA) distributes the request to other DSAs. This distribution is performed by either the DUA or the bind DSA, depending on how the service control options are set in the request. The service control options also determine how the request is handled during the distribution process.

1.6.3.1 Continuation Reference

A continuation reference describes how a service request can be continued at one or more other DSAs. It contains access points which consist of a DSA name and its corresponding PSAP address. A continuation reference is typically created when a DSA is unable to fulfill the request itself.

The continuation references are returned to the DUA either as a referral or as components of a partial outcome qualifier in the result of a **list** or **search** operation.

The DUA may use the access points, to bind the referenced DSAs and to forward the request to them.

Refer to Appendix I, ‘‘Navigation in GDS’’, for a more detailed description of how GDS handles continuation references.

1.6.3.2 Service Controls

Service controls are passed from DUA to DSA using DAP protocol. If the DSA chains to other DSAs, the service controls are passed unchanged to these other DSAs.

When service controls are enabled, they affect how a Directory Service request is distributed as follows:

LOCAL SCOPE
The evaluation of the request is restricted to the bind DSA. No references are generated and the distribution of the request is not performed. If this service control is disabled, the DSA can generate continuation references.

CHAINING PROHIBITED
No chaining is done in the DSA. References are returned to the DUA.

AUTOMATIC CONTINUATION (known only in the DUA)
If the DUA gets continuation references from the DSA, they are handled in the DUA. The DUA will send the request to the referenced DSAs. If this service control option is not set, the references are returned to the user or application program that initiated the request.

PREFER CHAINING
The bind DSA handles the continuation references and forwards a chained request to the referenced DSA (**LOCAL SCOPE** and **CHAINING PROHIBITED** service options must be disabled).

In the case of a **LIST** or **SEARCH** operation, the request may be decomposed and subrequests are forwarded to the referenced DSAs. Even if **PREFER CHAINING** is set, the DSA may still end up with a set of unresolved continuation references which it has to pass back to the DUA.

Chapter 2

GDS Components

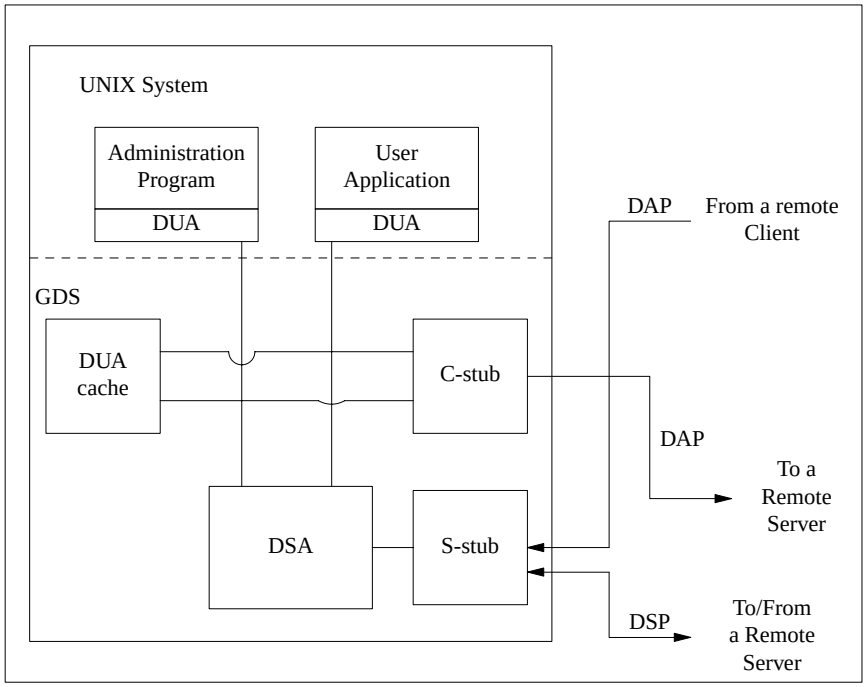
This chapter describes the components of GDS and how they work.

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, with 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.

The following figure 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 later in this chapter). 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 Application Program Interface (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 package and the Message Handling Systems Directory User package provide additional information objects for use by security, cache management, and electronic mail applications when using GDS.

(Refer to the *DCE 1.2.2 Application Development Guide* for more information on programming with XDS API.)

2.3 GDS Client/Server Communication

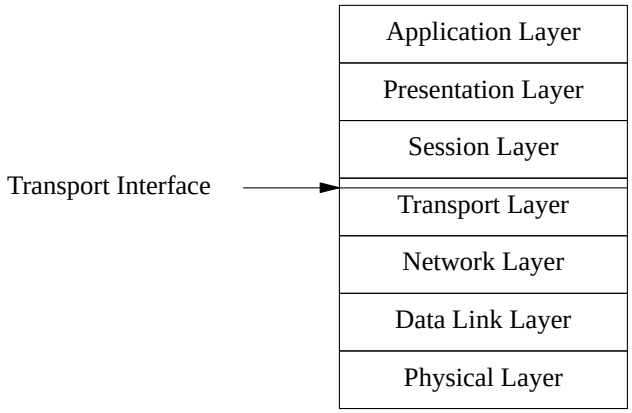
The X.500 directory service standard is written with a view to running on top of the Open Systems Interconnection (OSI) communications protocols. The OSI protocols are divided into seven layers:

- 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 double line 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 (either OSI transport or TCP/IP transport). 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), so GDS provides the capacity for running over the TCP/IP transport protocol as specified in RFC 1006. OSI transport services are accessed via the XTI-interface; the TCP/IP transport service is accessed via the socket interface. All transport services available within a particular system can be used in parallel.

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 Presentation Service Access Point (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 nonGDS service may interpret the P-Selector and S-Selector fields.

The following figure shows how a server address is entered in Mask 7a.

Figure 2–3. Example of a Server Address

```
(Mask 7a)                          DIRECTORY SYSTEM                          operation


P-Selector:                  □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
S-Selector:                  □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
T-Selector:                  Server

NSAP-Address 1:                   TCP/IP!internet=192.35.18.2+port=21011

NSAP-Address 2:                   IBMLAN!ethernet=080014151475

NSAP-Address 3:              □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
NSAP-Address 4:              □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
NSAP-Address 5:              □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

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 Local Area Network (LAN) address used on IBM RS6000 machines.

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 provide 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

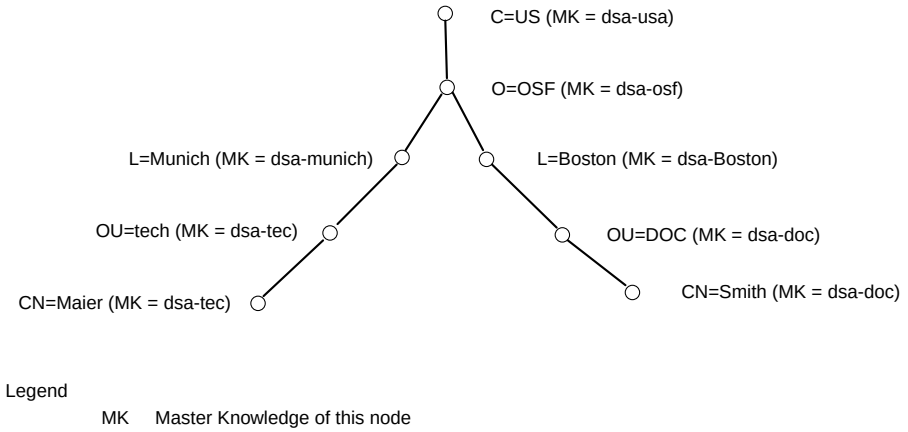
Each DSA must contain the schema object under the root of the DIB. If a DSA has the master entry of the schema object, it is called an initial DSA. If other DSAs contain a shadow of the schema from an initial DSA, these DSAs constitute an administrative domain. To make a DSA part of an administrative 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 nonGDS DSAs, administrators also need to make sure that the shadows of these nonGDS 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 of these remote DSAs, the administrator must select the **Logon to a Specific DSA** option in Mask 1. (Refer to 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 DNs 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 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 corporate 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 ensuring 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 hexadecimal 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 **cdscpshow 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**:

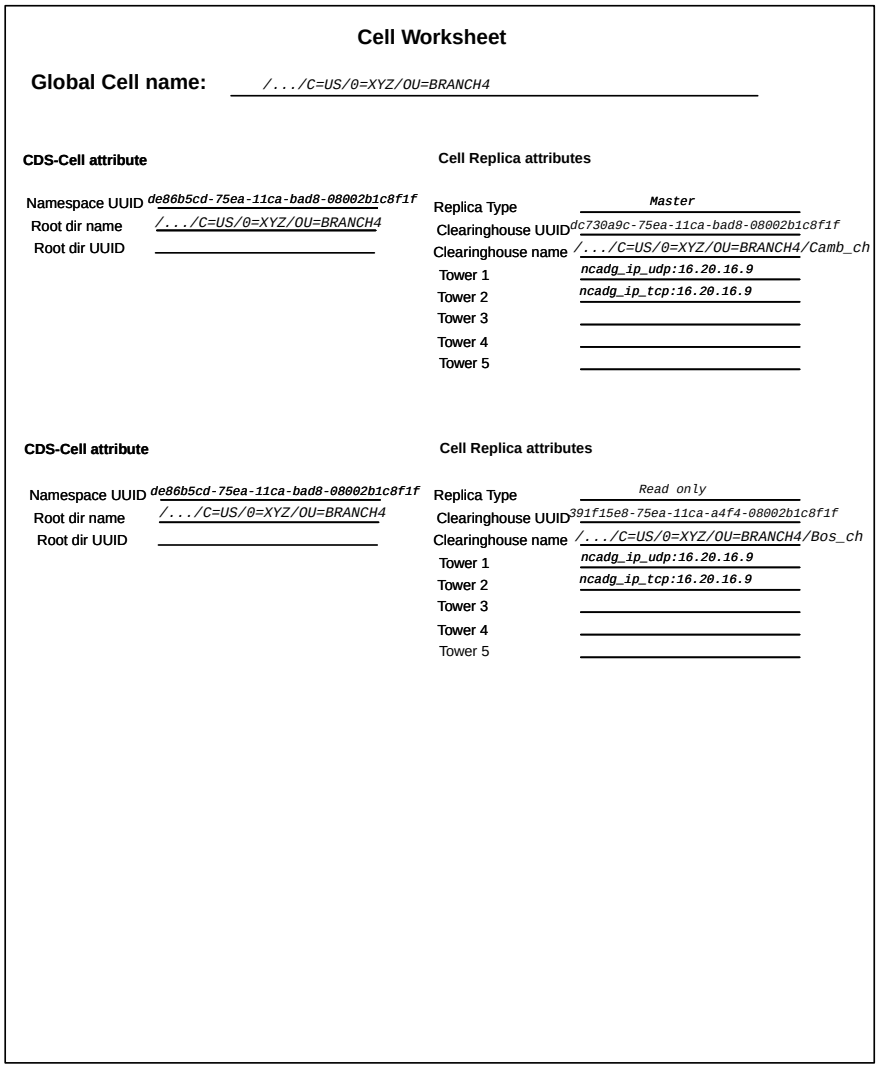
```
cdscp> show cell /.../C=US/O=XYZ/OU=Branch4 as gds


               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/0=XYZ/OU=BRANCH4

CDS-Cell attribute

Namespace UUID de86b5cd-75ea-11ca-bad8-08002b1c8f1f
Root dir name /.../C=US/0=XYZ/OU=BRANCH4
Root dir UUID

Cell Replica attributes

Replica Type Master
Clearinghouse UUID dc730a9c-75ea-11ca-bad8-08002b1c8f1f
Clearinghouse name /.../C=US/0=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/0=XYZ/OU=BRANCH4
Root dir UUID

Cell Replica attributes

Replica Type Read only
Clearinghouse UUID 391f15e8-75ea-11ca-a4f4-08002b1c8f1f
Clearinghouse name /.../C=US/0=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.

These access classes are not related in a hierarchical order; they are independent. For example, if a user has modify rights for sensitive attributes, this does not imply access rights for standard attributes. Each access right must be explicitly assigned.

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.

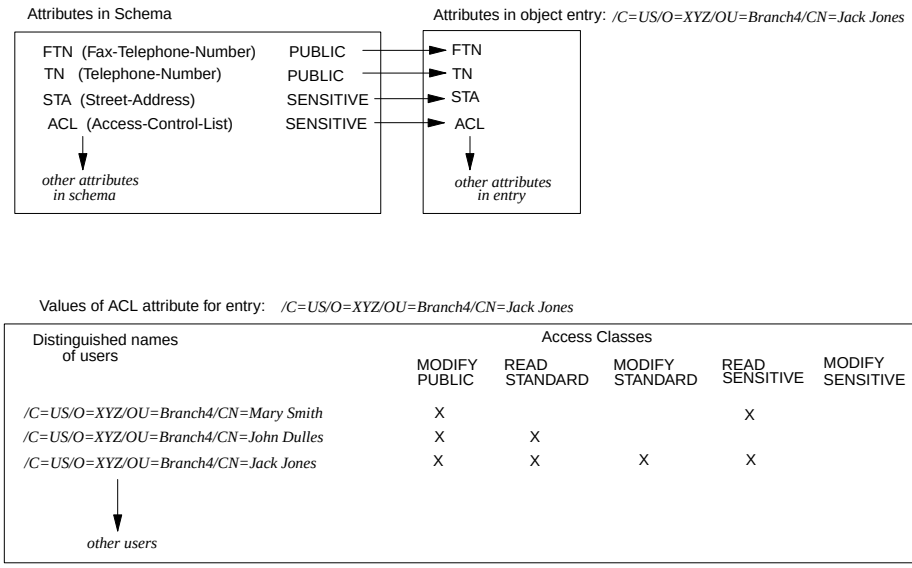
John Dulles has **MODIFY PUBLIC** and **READ STANDARD** access. This means that

- 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



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.

3.3.1 ACL Worksheets

There are two types of ACL worksheets:

- ACL Schema worksheet
- ACL Object Entry worksheet

3.3.1.1 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

3.3.1.2 ACL Object Entry Worksheet

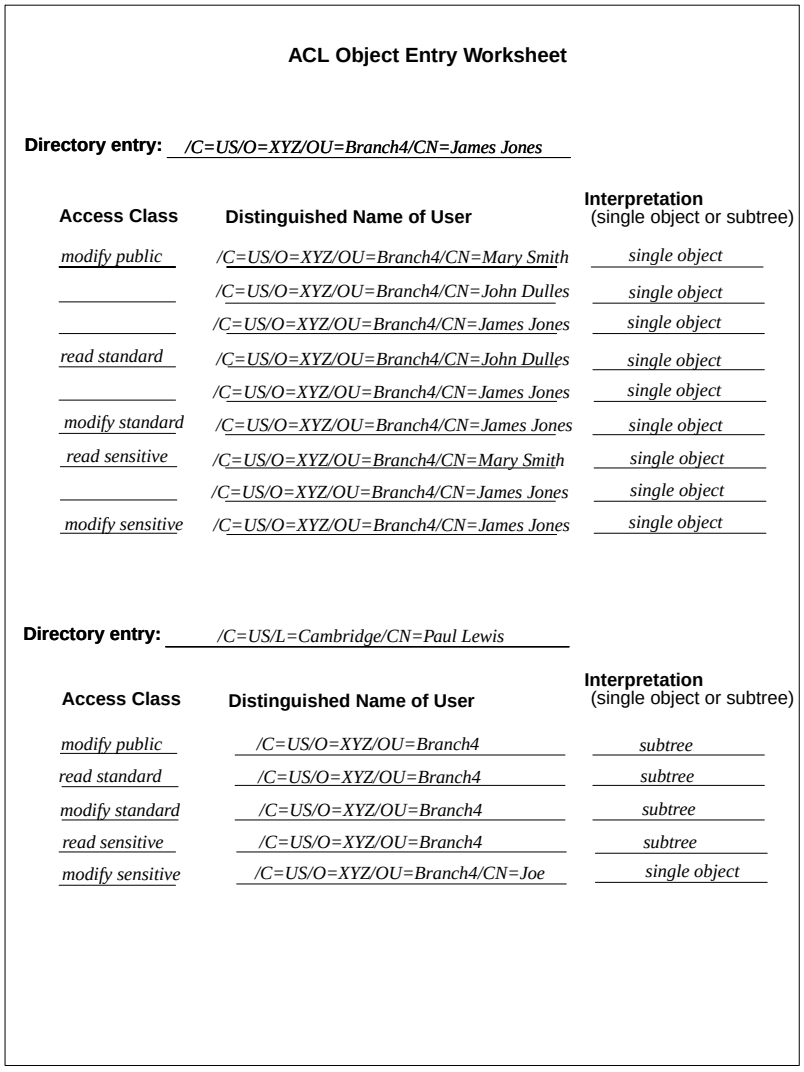
The ACL Object Entry worksheet contains a list of the DNs and access classes assigned to each user for each object entry that requires specific access to its attributes. The information in the worksheet specifies the values of the ACL attribute for a specific object. The **Interpretation** heading indicates 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*

3.3.2 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.

3.3.3 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**. **dsa-employ** masters all employee records for the branch; the **dsa-ops** DSA 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 United States). 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-Typedefault**). 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. For more information on how to apply for an NSAP address, refer to Appendix D.

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 Updating the DCE Registry

Administrators are required to specify the authentication mechanisms supported by a DSA when configuring the directory ID of the Directory System. GDS supports anonymous, simple authentication (requiring a user's DN and a password) and DCE authentication.

Administrators can use DCE authentication as a method to provide additional security in accessing DSAs in the DCE environment. Users defined as principals in the DCE registry can be permitted or restricted access to the Directory Service. These restrictions can apply to XDS applications that bind to specific DSAs during program execution, to users logging into DSAs via the GDS administration program's login facility and **gdscp**, and to DSAs communicating with other DSAs in the DCE environment.

Refer to the *DCE 1.2.2 Application Development Guide* for more information on applying security features within an XDS application program. Refer to Chapter 7 for more information on how DCE authentication affects user login to a DSA.

The DCE Security Service provides a higher level of security than Simple authentication mechanisms for the transmisssion of a user's identity (DN) to the DSA during a **bind** operation. You must enter the DNs of directory users and DSAs to the respective principals in the DCE registry to make DCE authentication work.

The DCE registry contains by default the following extended attributes necessary for GDS if DCE authentication is enabled:

X500_DN Contains the X.500 DN of a principal (user or DSA)

X500_DSA_Admin
Contains a list of DSAs to which the principal is allowed to perform a master bind

The **X500_DSA_Admin** attribute is only necessary for adminstrators who are principals and only if the administrator of the DSA intends to specify shadow update jobs. In order to perform shadow updates, **gdsditadm** binds to all other DSAs that hold shadows of information mastered by the local DSA. You must specify the DNs of these other DSAs in the **rgy** attribute **X500_DSA_Admin** of the administrator principal running **gdsditadm**.

Use the DCE **dcecp** command to assign a DN to existing principals. Only principals with management permissions can add or modify the values for the **X500_DN** and **X500_DSA_Admin** attributes.

For example, suppose a user logs into to the DCE as the principal **joe** and wants to access the Directory as the X.500 user with DN of **/C=de/O=sni/OU=nm12/CN=joe,CN=miller**. The administrator assigns the DN to the DCE principal **joe** by using the following **dcecp**command:

```
dcecp>principal modify joe -add {X500_DN {/C=de/O=sni/OU=nm12/CN=joe,
CN=miller}}
```

Refer to the *DCE 1.2.2 Command Reference* for information on how to use **dcecp**.

Note that the example implies that the principal **joe** is already known to the Registry. Furthermore, the user represented by **joe** needs an account to **dce_login** before he can benefit from DCE authentication within GDS applications. This will most likely be the case because, typically, users of DCE do not exclusively use the Directory Service.

If the user does not have an account and principal (which is most often true for DSAs because they have to be introduced in the DCE Registry only for the purpose of authentication), the Registry administrator must:

1. Choose a principal name (in the example it is **gds-dsa1**). This name gets into the Directory as well as the **PN** attribute of the DSA. The **PN** attribute must be the global dce name and take the form:

   ```
   /.../cell-name/dsa-principal-name
   ```

2. Ask the administrator who maintains the DCE cell for this principal to apply the following **dcecp** commands:

   ```
   dcecp>principal create { gds-dsa1 }
   dcecp>principal modify gds-dsa1 -add {X500_DN {/C=re/O=sni/OU=buba
   /CN=dsa/CN=dsa1}}
   dcecp>group add groupname -member { gds-dsa1 }
   ```

```
dcecp>organization add orgname -member { gds-dsa1 }
dcecp>account create gds-dsa1 -group {groupname} -organization
{orgname} -password {gds-passwd}
dcecp>keytab create keytabname -attr { {storage
/opt/dcelocal/gds_keystore} {data {gds-dsa1 plain 1 gds-passwd}}}
```

3.7 Creating and Maintaining the Trusted DSA Table

The administrator can restrict authenticated access by specific DSAs by creating and maintaining the Trusted DSA Table (TDT). These restrictions apply if a chained operation follows a bind initiated by the intermediate DSA via the DCE authentication mechanism. Each table entry contains the DN of a DSA and permission tags that define the type of access rights permitted for the DSA.

The following permission tags can be set for each DSA in the table:

r The **read**, **list**, **search**, and **compare** operations chained by the DSA are treated as if the originator had performed a DCE authenticated bind. All other operations are treated as if an anonymous bind had been performed.

m The **modify entry**, **modify RDN**, and **compare** operations chained by the DSA are treated as if the originator had performed a DCE authenticated bind. All other operations are treated as if an anonymous bind had been performed.

rm All operations are treated as if the originator had performed a DCE authenicated bind.

If the administrator does not specify a permissions tag for a TDT entry, all operations chained by the DSA are treated as if an anonymous bind had been performed.

The administrator creates the file by using a text editor and saves it in a file named **/opt/dcelocal/var/directory/gds/dsa/dir***/dir-id* **/dsatdt**. For example, if the administrator wants to create a TDT file for the DSA configured with a directory ID of **1**, the pathname of the TDT file should be: **/opt/dcelocal/var/directory/gds/dsa/dir1/dsatdt**.

The administrator of the DSA **/C=de/O=sni/OU=buba/CN=dsa1** might set up the TDT file as follows:

```
/C=de/O=sni/OU=buba/CN=dsa,CN=dsa99:rm
/C=de/O=siemens/OU=zfe/CN=dsa05:m
/C=us/O=osf/OU=dce1.1/CN=gds_dsa:r
/C=iq/O=abc/OU=xyz/CN=suspect_dsa:
```

After this sample TDT is created, if the target DSA receives a DSP bind request via the DCE authentication mechanism, the TDT check is switched on. The intermediate DSA now chains the operation. The chained operation request contains the name of the client (the X.500 user who bound the intermediate DSA). This name is referred to as the originator name. The originator name is the only name that is used by the target DSA to perform access control checks by using the ACL attribute of the object involved in the operation.

The following occurs depending on which DSA is the intermediate DSA:

/C=de/O=sni/OU=buba/CN=dsa/CN=dsa99
: The originator name is used for access control decisions in every type of operation. This means that the result of the access control check is the same as if the client had bound and authenticated directly to the target DSA.

/C=de/O=siemens/OU=zfe/CN=dsa05
: The originator name is used for access control decisions in modify entry, modify rdn and remove entry operations. This means that the result of the access control check for other operations is the same as if the client had bound anonymously to the target DSA.

/C=us/O=osf/OU=dce1.1/CN=gds_dsa
: The originator name is used for access control decisions in read, list search and compare operations. This means that the result of the access control check for other operations is the same as if the client had bound anonymously to the target DSA.

/C=iq/O=abc/OU=xyz/CN=suspect_dsa
: The originator name is *not* used for access control decisions in any operation. This means that the result of the access control check for every operations is the same as if the client had bound anonymously to the target DSA.

A DSA not contained in the TDT file list
: The originator name is *always* used for access control decisions in any operation. This means that the result of the access control check for

every operation is the same as if the client had bound and authenticated directly to the target DSA.

If the TDT file does not exist, there is no restriction on chained operations, except that the following message appears in the log file: `no TDT available` .

The administrator can switch on the TDT feature by creating a TDT file and entering the DSA that he or she does not trust to fill in originator names correctly.

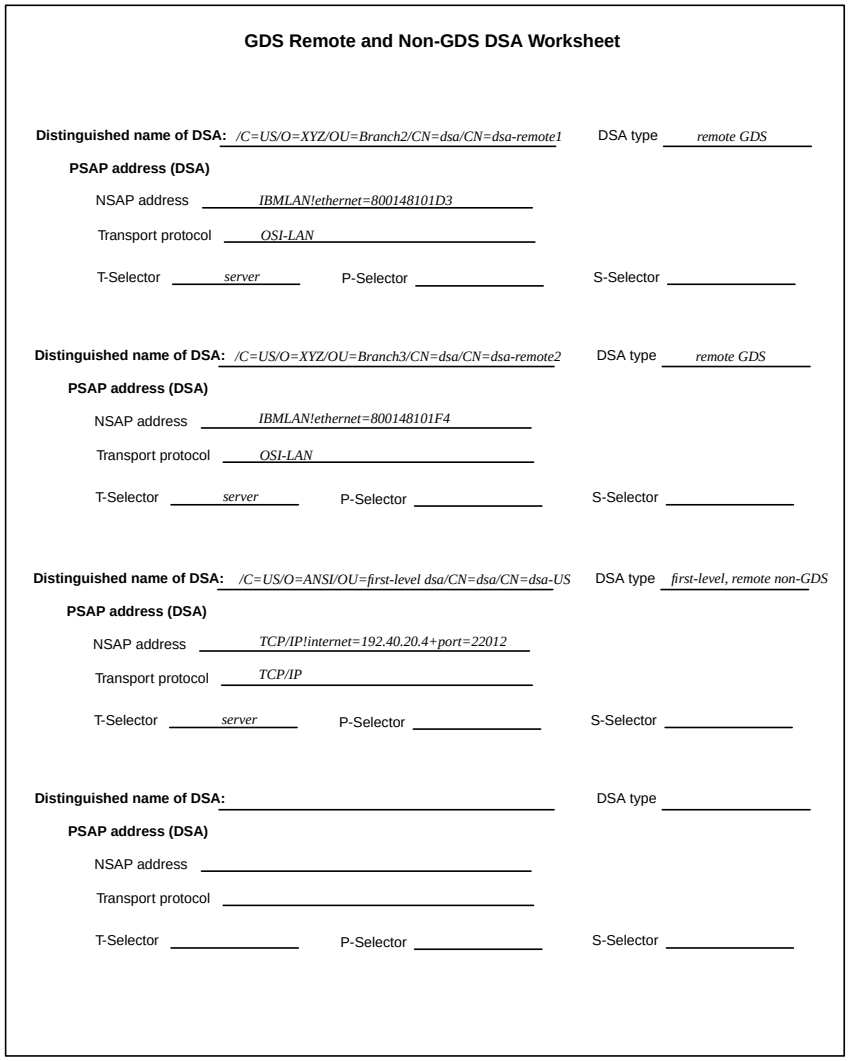
3.8 Defining Remote GDS and Non-GDS DSAs

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 administration 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 administered 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 provided as part of the GDS software. The administrator uses batch mode to run shell scripts that have been created to automate administrative procedures. Refer to the *DCE 1.2.2 Command Reference* for more information on how to use GDS commands in shell scripts.

The GDS command-line options are described in the *DCE 1.2.2 Command Reference*.

You can also use the **gdscp** command-line interface as an alternative to the object administration menu-interface and part of the cache administration menu-interface. See Section 4.3 for a description of object administration and cache administration.

GDS can be maintained by using the following commands:

gdsdirinfo Obtains information on all daemon processes running for GDS and on all current processes using GDS Provides or removes the IPC resources (shared memory, message queue, and semaphore) that are used by GDS to allow the communication between the different GDS components (such as GDS-applications (DUA), DUA cache, C-stub, S-stub, and DSA)

gdsipcstat Used by an administrator to localize and isolate GDS problems.

gdssetup Provides the administrator with an interface to simplify the process of creating and initializing a directory configuration

gdsstep Evaluates a trace file containing a GDS trace

These commands are not described in this chapter. See Chapter 5 for a description of the **gdsipcinit**, **gdsipcstat**, and **gdsstep** commands. See the reference pages for descriptions of the other maintenance commands.

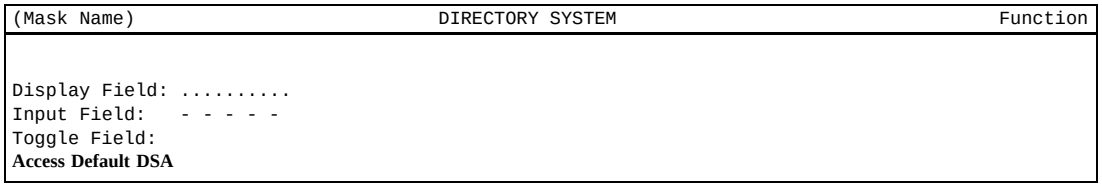
4.1 Mask Structure

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

- Current mask name
- 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 this manual, 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 **l** 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 object administration functions, 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, nonGDS DSAs).

Table 4-1 shows functions that are supported for logging into to 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 value of an attribute. 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 a leaf 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.

Function	Meaning
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.
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.

Function	Meaning
Display Shadowing Jobs	Displays existing shadowing jobs.
Display Update Errors	Displays errors which occurred during shadow updates.
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 save, 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.

Function	Meaning
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 gdscacheadm

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) of **gdsditadm**. cache administration supports object administration 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 on to 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	**Functionality**
F1	Returns from attribute masks to object list (Mask 18) in the case of **Display Objects**. Goes from object list (Mask 18) to object name mask (Mask 6) in the case of **Display Objects**. Goes from O/R-Name mask (Mask 34) to O/R-Address masks (Masks 28, 29, 30, 31, 32) in the case of **Modify Attribute**. Goes from O/R-Address (Mask 32) to PSAP mask (Mask 7a) in the case of **Modify Attribute**.
<Return>	Goes to next field/line.
<Menu>	Executes the function.
<Scroll Up>	Displays the previous page/object/entry.
<Scroll Down>	Displays the next page/object/entry.
<Del Char>	Deletes a character.
<Del Line>	Deletes a line.

Key Name	Functionality
<Ins Char>	Inserts a character.
F8	Selects a recurring attribute in Mask 6d.
<Del>	Aborts function and return.
<Ctrl-C>	Aborts function and return.
<End>	Terminates **gdssysadm**.
Ctrl-D>	Terminates **gdssysadm**.
<Help>	Displays Help mask.
Cursor Right	Navigates in the mask.
Cursor Left	Navigates in the mask.
Cursor Up	Navigates in the mask.
Cursor Down	Navigates in the mask.
<Backspace>	Deletes a character.

Note: The first column in Table 4-6 corresponds to keys available on standard Siemens/Nixdorf Information Systems (SNI) keyboards. See Section 4.6 if you do not use a standard SNI keyboard.

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

- Insert a character at the cursor position by using **<Ins Char>**.
- Delete a character at the cursor position by using **<Del Char>**. The character before the cursor position can be deleted 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.

Confirm the input to the mask by either pressing **<Menu>** anywhere in the mask or by pressing **<Return>** in the last input field of the mask.

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>** or **<Ctrl-C>**. 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 in selection lists.

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.

You can press the**<HELP>** key at any time while a GDS administration program is running to display one or more help screens. The help screens describe the purpose of the current mask, provide brief descriptions of the fields and parameters, ranges of input values, and other information contained in the mask.

Help text is displayed in a series of scrolling screens. Press **<CR>** to display the next screen of information. When the last screen of help text is displayed, press **<CR>** to exit from Help.

4.6 Code Set Mapping

User input is expected by the GDS administration programs in **8859-1** (**Latin-1**) code. The GDS administration programs do an automatic mapping to the **T.61** code set. Mapping problems may occur from **8859-1** to **T.61** or from **T.61** to **8859-1** code sets. A question mark (?) is displayed for **T.61** characters that cannot be displayed in **8859-1** code. If you enter **8859-1** characters that cannot be mapped to **T.61** character codes, an error message is displayed.

4.7 Keyboard Mapping

The GDS administration programs try to map each function key to a predefined termcap capability. Refer to Table 4-7 for the default mapping of function keys to

terminal capabilities. If one of these predefined termcap capabilities is not defined for your terminal, then the **adm_term_cap** file is used to map missing capabilities to existing ones. For example, the **Menu** key can only be redefined if termcap capability **ya** is not defined in the termcap or terminfo file.

The **adm_term_cap** file is located in the directory **/opt**/*dcelocal***/var/adm/directory/gds/adm**.

You can use the mapping of another file if you set the environment variable **GDS_TERM_CAP** with the pathname and filename. If **gdsditadm** cannot open this file, it will use the default file by the name of **/opt**/*dcelocal***/var/adm/directory/gds/adm/adm_term_cap**.

Table 4–7. Default Terminal Capability Mapping for the GDS Administration Tools

Key Name	Symbolic Name Used in adm_term_cap	Termcap Capability
F1	**FKTF1**	**k1**
F8	**FKTF8**	**K8**
<Help>	**FKTHELP**	**l0**
<Return>	**#FKTCR**	
<Menu>	**FKTMENU**	**ya**
<Scroll Up>	**FKTSCU**	**kR**
<Scroll Down>	**FKTSCD**	**kF**
<Cursor Right>	**FKTCURS_RIGHT**	**kr**
<Cursor Left>	**FKTCURS_LEFT**	**kl**
<Cursor Up>	**FKTCURS_UP**	**ku**
<Cursor Down>	**FKTCURS_DOWN**	**kd**
<Backspace>	**FKTBS**	**kb**
<Del Char>	**FKTDELCHAR**	**kD**
<Del Line>	**FKTDELLINE**	**kL**
<Ins Char>	**FKTINSCHAR**	**kI**

Key Name	Symbolic Name Used in adm_term_cap	Termcap Capability
<Del>	**#FKTDEL**	
<Ctrl-C>	**#FKTDEL**	
<End>	**#FKTEND**	

Note: GDS administration program capabilities preceded by a pound sign (#) are processed directly by the GDS administration program. These entries are described later in this section.

For redefinition of a terminal capability you have to edit the **adm_term_cap** file. Replace the names of termcap capabilities that are undefined in your specific machine/ terminal configuration by valid ones. For example, if **ya** is not defined but **k2** is, it could be replaced by **k2**. Function key **F2** would then act as the **<Menu>** key.

The exact format of **adm_term_cap** is described in the file itself.

4.8 Administration of GDS By 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 that remains 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 **:*** (colon 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
:Authentication mechanism to be used:Simple unprotected
: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:Country
: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 **dced** 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 (**dced**)
  - Security (**secd**)
  - 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 *DCE 1.2.2 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).

The trace system for other processes uses the OSF serviceability. Each GDS daemon process and each XDS application may generate log files containing messages with the following severity levels:

- FATAL
- ERROR
- WARNING
- NOTICE
- NOTICE VERBOSE

This type of logging is called *Logging for Exception Handling*. All of these log files contain human-readable messages. The routing of the Logging for Exception Handling describes which severity levels should be logged, and where the messages should be written to. The logging may be specified in detail by the use of **dcecp**. The meanings of these messages and actions, which may be performed by an administrator on occurrence of these messages, are described in the *DCE 1.2.2 Problem Determination Guide*.

Another log file can be generated by each GDS daemon and each XDS application, which contains just traces in a binary format. It can be evaluated by the GDS trace evaluation program **gdsstep**. Refer to the **gdsstep** reference page for detailed information on how to use **gdsstep**.

The routing of the logging for traces describes which subcomponents should log, the debug level for each such subcomponent, and where the messages should be written to. The logging can be specified in detail by the use of **dcecp**. The messages are self-explanatory and describe the activities of the programs. The GDS daemons and applications contain the following subcomponents:

general	General GDS logging
ipc	Interprocess communication
apdu	Application program data units
asn1	Abstract Syntax Notation 1
ros	Remote operation service
cmx	Communication Method SINIX
pfm	Performance logging

The routing of any application (including **gdscacheadm** and **gdsditadm** when called from the command line) follows the specification of the shell variables **SVC_***severity*

and **SVC_GDS_DBG** , which are described in the **svcenv(1)** reference page. Note, that **SVC_GDS_DBG** may specify only a **BINFILE** logging.

When the trace system is activated by the directory system administration, the Logging for Exception Handling for each severity level as well as the logging for traces is enabled with default routes.

The logging for exception handling is routed into TEXTFILEs. The messages of all severity levels are written into one file. Each file contains at most 100 log entries.

The logging for tracing is routed into BINFILEs. Each file contains at most 2000 entries.

When the first log file is full, a second log file is generated with another suffix. When the second log file is full, the first log file is overwritten.

Table 5-1 describes where these log files are located.

Table 5–1. Log File Subdirectories

Process	**Subdirectory**
gdsditadm	*dcelocal*/**var/adm/directory/gds/dna**
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/dsa/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 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.

The name of the logfile for exception handling is: **EXC** *pid.suffix*

The name of the log file for traces is: **LOG***pid.suffix*, where *pid* represents the process ID and *suffix* is 1 or 2.

The following sections describe the meaning of the debug levels in serviceability calls of the various subcomponents.

5.5.1 general

For **general**, the default debug levels for various processes are the following:

Debug level 1
: **gdsditadm**, **gdscache**, **stubs**, **gdsipcchk**, and **gdsdaemon**

Debug level 2
: **gdsdsa**, **gdsmkiss**, **gdsgendb**, **gdstransfer**, and **gdsmkupd**

Debug level 1 contains the following:

- All messages that are issued in case of failing system calls or C-ISAM calls.
- All messages that are issued in case of error events (such as illegal operation ID received, illegal distributed command, and so on).
- In **gdsdsa**, **gdsmkiss**, **gdsgendb**, **gdstransfer**, **gdsmkupd**, and **gdsdaemon**, all messages for exception handling are written into the log file with debug level 1, all messages from a booting or restarting DSA, and all messages that indicate entry or exit of an operation.
- In **gdscache**, activation and deactivation of a directory-ID specific DUA cache, DUA cache configuration information, and DN verification problems.
- In **gdscstub** and **gdssstub**, internal stub activities.
- In **gdsipcchk**, internal IPC-verification activities.

Debug level 2 contains the following:

- In **gdsdsa**, all messages that describe process handling and central points in the processing of requests, such as
  - — Result of local name resolution
  - — Searching objects in database

  - — Searching of references
  - — Alias dereferencing
  - — Chaining
  - — Code points of performance logging
  - — Shadow update logging
- In **gdsgendb**, all messages that describe central points in the processing of requests, such as:
  - — Reading of configuration data
  - — Reading of components of the schema object
  - — Processing of each table of the schema
  - — Creation of each C-ISAM file and scheme file
- In **gdstransfer**, all messages that describe central points in the processing of requests, such as
  - — Result from comparing the schemes
  - — Finding an object record
  - — Successful transfer of an object record
- In **gdsmkupd**, all other messages.
- In **gdscache**, internal DUA-cache activities.
- In **gdscstub** and **gdssstub**, DUA cache update activities.

Debug level 3 contains the following:

- In **gdsdsa**:
  - — Function entry and exit logging for internal functions
  - — Logging of each found entry
  - — Logging of each dereferenced alias
  - — Logging of each found access point
  - — Logging of each used C-ISAM index
  - — Detection of user errors that result in an error message

- In **gdsgendb**, reading of each table entry of the SRT, OCT, and AT.
- In **gdstransfer**, function entry and exit logging for internal functions and logging of the occurred errors in transforming a record.
- In **gdscstub** and **gdssstub**, TP-routines activities.
- In **gdsdaemon**, function entry and exit logging for internal functions and all messages that describe central points in the processing of the shadowing job to be performed.

Debug level 4 contains IPC-/network-event management activities in **gdscstub** and **gdssstub**.

Debug level 6 contains the following:

- In **gdscache**, additional internal DUA cache activities.
- In **gdsipcchk**, named FIFO handling.

5.5.2 apdu

For **apdu**, the default debug level for **gdsditadm**, **gdsdaemon** and **gdsdsa** is 1. The default debug level is inactive in all other processes.

5.5.3 pfm

For **pfm**, this logging is available in the DSA to a limited extent. It is inactive on default.

5.5.4 ipc

For **ipc**, the default level is 2 for all processes.

Debug level 1 contains function entry and exit logging of all IPC-interface function calls including parameters.

Debug level 2 contains all IPC-messages sent and received by IPC through message queue (without message data) and all attention data sent and received through named FIFOS.

Debug level 3 contains additional named FIFO handling.

Debug level 4 contains function entry and exit logging of internal IPC-function calls.

Debug level 6 contains all IPC-message data sent and received through shared memory.

5.5.5 ros

For **ros**, the default debug level is 1 for the Stubs process and inactive for all other processes.

Debug level 1 contains function entry and exit logging of all ROS interface function calls.

Debug level 2 contains function entry and exit logging of all transport interface function calls.

Debug level 3 contains internal logging of protocol machine activities.

The default debug level is 1 for the Stubs process and inactive for all other processes.

Debug level 1 contains function entry and exit logging of all transport interface function calls and internal transport provider (socket/XTI) interface activities. Debug level 3 contains function entry and exit logging of NDS interface calls.

Debug level 6 contains internal NDS activities.

5.5.6 asn1

For **asn1**, the default debug level is 2 for the **gdsdsa** and **gdscstub** processes. It is inactive for all other processes.

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.

5.5.7 Displaying Status By Using gdsdirinfo

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 5-2.

Table 5–2. 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 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 5-3 shows the valid states and their meanings.

If none of the specific states shown in Table 5-3 is associated with the process, a dash is printed.

Table 5-3 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 5–3. 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** process type 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 the GDS is inactive, the following message is written to **stderr**:

```
A shmget system call has failed (key = 1148635637,\
access mode = 666, errno = 2).
```

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

For a detailed description, see the **gdsdirinfo** reference page.

5.5.8 Interpreting Log Files By Using gdsstep

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.

For a detailed description, see the **gdsstep** reference page.

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

Use one of the following programs to configure and initialize the Directory Service:

gdssysadm A menu-driven interface that requires the administrator to have a complete knowledge of the initialization steps and to perform each step manually or use a batch file script.

gdssetup A simplified command-line or interactive interface that performs the initialization steps automatically and requires only a small set of configuration parameters as input. The command line interface allows the administrator to create an input file that can be modified and reused to try out different configurations. The interactive interface consists of a series of prompts to which the administrator supplies input values.

Refer to the **gdssetup** reference page for a detailed description of how to use **gdssetup** to configure and initialize the Directory Service. Note that it is recommended that you read the information on intialization steps in this chapter before attempting to use **gdssetup**. Also, you must use **gdssysadm** to activate the Directory Service after you have completed the initialization.

To perform configuration and initialization by using **gdssysadm**, run **gdssysadm** by entering the following at the system prompt:

```
sysprompt>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
    Authentication mechanism that the DSA supports ?:
Simple Unprotected
```

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:

-
-

The following items are only displayed if you select **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.

Authentication mechanism that the DSA supports?
Select one of the following fields by pressing the space bar:

Simple Unprotected
Authentication based on DN and password of user.

DCE Authentication
Authentication based on DCE security. Users must be registered as principals in the DCE Registry and must be given the extended Registry attribute **X500_DN**.

Simple and DCE
Authentication based on DN, password of user, and DCE security. Users must be registered as principals in the DCE Registry.

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.

Note: 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  AUTH-MECH
1          Clt/Srv-System         1           16      no      SIMPLE
3          Clt/Srv-System         1           16      no      SIMPLE


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 5-1). 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 the 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).

If DCE authentication is used as the authentication method and if the DSA is local, an additional mask is displayed so that the administrator can add the principal name of the DSA.

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.

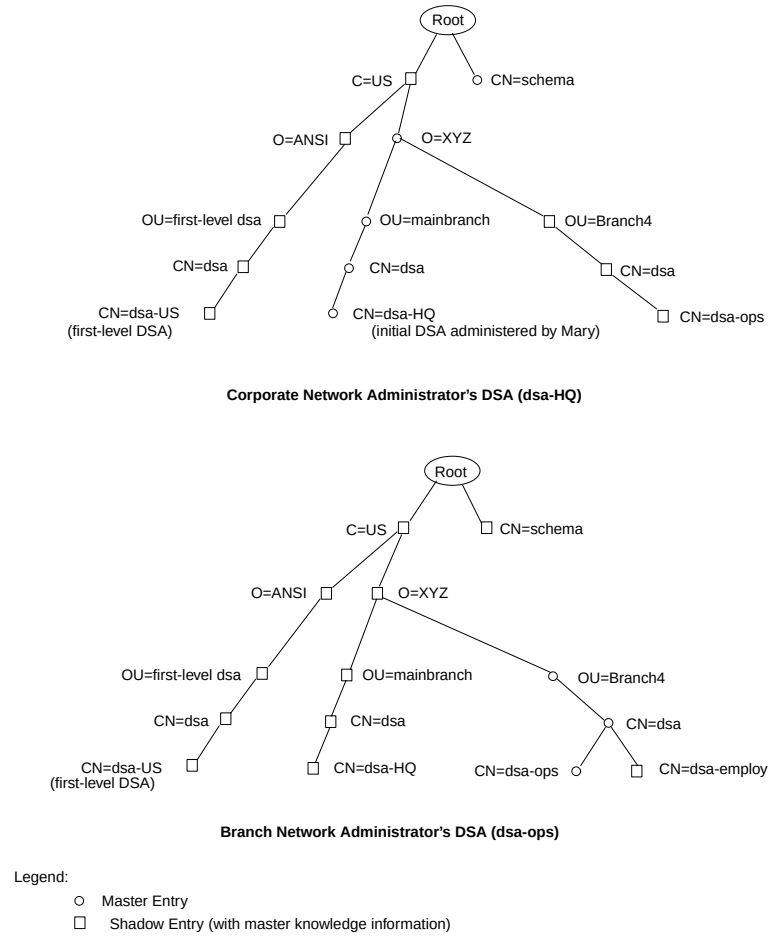
The values for **DSA-Type** that the branch network administrator entered on the sample Client worksheet and Client/Server worksheet are shown in Figures 3-6 and 3-7.

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



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, which is **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 nonGDS 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 (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)
- If DCE authentication is used as an authentication method, the principal name of the local DSA

The following client/server information is required:

- Client address (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.
- If DCE authentication is used as an authentication method, the principal name of the local DSA

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 nonGDS DSAs
- A client/server system with DSAs that do not constitute an administrative domain
- An administrative domain using 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 nonGDS 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 nonGDS 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 That 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 That 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 by using the new schema structure (step 6).
7. Enter the initial DSA (including its PSAP address) and its principal name (if DCE authentication is supported) in the DUA cache by 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 That Uses the Default Schema

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

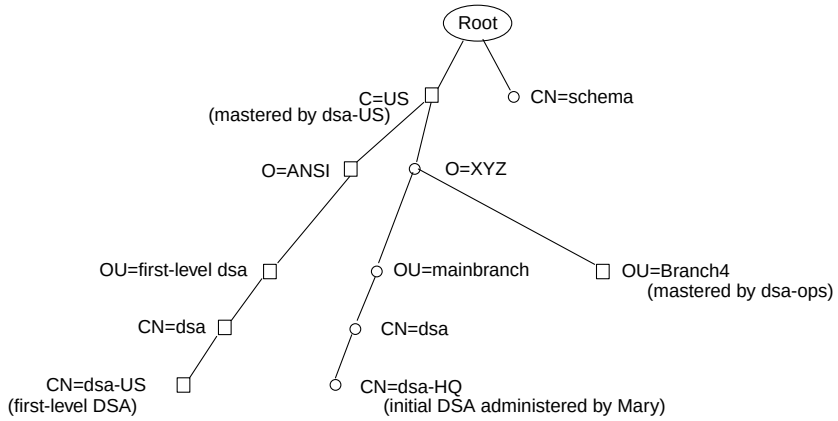
This sample initialization demonstrates how Branch Administrator 4 adds a DSA to an administrative domain and creates the subtree of the entry **/C=US/O=XYZ/OU=Branch4** on**dsa-ops** so that it is mastered by **dsa-ops**.

Before starting the initialization of **dsa-ops**, Branch Administrator 4 tells Corporate Administrator 1 to create a shadow of the entry, where:

- **/C=US/O=XYZ/OU=Branch4** is the DN of the entry for which the shadow is to be created.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of Branch Administrator 4's DSA, which will be the shadow's **Master-Knowledge** attribute.

No ACL will be set by Corporate Administrator 1. The following figure 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



Legend:

- ○ Master Entry
- □ Shadow Entry (with master knowledge information)

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=192.35.189.4+port=21018**

Step 2: Enter the local DSA, including it's 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 (in Masks 6d, 7, and 7a) the PSAP address of the local DSA, which is the same as the client address in step 1, but with a different T-Selector:

   TCP/IP!internet=192.35.189.4+port=21017

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**) with the DN of **dsa-HQ**:

   /C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops is the DN of the local DSA.

   /C=US/O=XYZ/OU=mainbranch/CN=dsa/CN=dsa-HQ is the full DN of **dsa-HQ**.

Step 5: **Enter a copy of the initial DSA object in the local DSA**

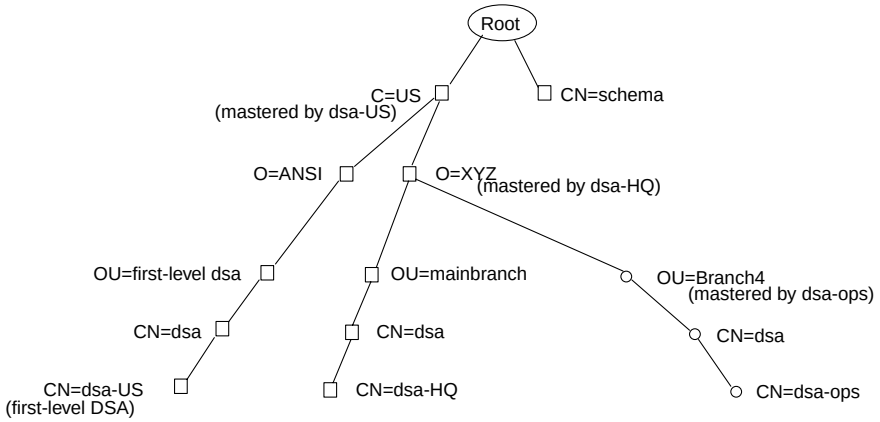
1. Branch Administrator 4 logs into the initial DSA by toggling the **Logon to a Specific DSA** option in Mask 1.

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 5 is 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



Legend:

- ○ Master Entry
- □ Shadow Entry (with master knowledge information)

The first time step 5 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 5 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 6: Enter local DSA as an object in the local DSA

1. Branch Administrator 4 creates the entry with DN **/C=US/O=XYZ/OU=Branch4**.
2. Branch Administrator 4 enters the local DSA, **dsa-ops**, as an object in the local DSA by using the sequence of steps in Section 6.3.4. Using Masks 3 through 7a, as described in **Step 10: Enter local**

DSA as an object in the local DSA in the following section, the branch network administrator adds the entry for **dsa-ops**.

Step 7: 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 in the following section.

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 to 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 log into to the DUA cache, and step 8 requires a login to the local DSA. After step 2 is performed, the administrator must log off from 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 value **local'** or **default/local'** for the attribute name **DSA-Type**. If DCE authentication is used as the authentication method, an additional screen is displayed so that the administrator can enter the principal name of the local DSA. Enter the global DCE principal name of the DSA in the format: */.../cellname/principal*.

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 by 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 value **local'** or **default/local'** for the attribute name **DSA-Type**. If DCE authentication is used as the authentication method, an additional screen is displayed so that the administrator can enter the principal name of the local DSA. Enter the global DCE principal name of the DSA in the format /.../*cellname*/*principal*.

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 to 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 the new parent node by entering the structure rule of the 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: For the attribute name **User-Password**, 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: For the attribute name **DSA-Type**, enter and the attribute value **default'**.

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

Step 10: Enter the 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 by 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

You can log into a DSA or the DUA cache by using one of the following methods:

- 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.
- Call the **gdsditadm** or **gdscacheadm** process directly.
- Use **gdscp** to issue a **bind** command.

7.1 Mask 1: User Identification

When the **Administration of the directory information tree** option is selected in the Menu Mask (Part 1), Mask 1a (Figure 7-1) is displayed. Enter the directory ID and the authentication method that you will use.

Figure 7–1. Mask 1a: Authentication Mechanism

```
(Mask 1a)                              DIRECTORY SYSTEM                              Logon

Directory ID:
1
Authentication mechanism to be used:
Anonymous
```

Mask 1a displays the following fields:

Directory ID
Enter the directory ID of the directory to be administered.

Authentication mechanism to be used
Select one of the following options by pressing the space bar:

- **Anonymous** (default)
- **Simple Unprotected**
- **DCE Authentication**

Press MENU or RETURN to display Mask 1b (part 2 of the logon mask) as shown in Figure 7-2.

Figure 7–2. Mask 1b: User Identification

```
(Mask 1b)                              DIRECTORY SYSTEM                              Logon


                                          Directory ID:
__
Authentication mechanism to be used:
Simple Unprotected
Password:
Country-Name:                 - -
Organization-Name:            - - - - - - - - - - - - - - -
Org.-Unit-Name:               - - - - - - - - - - - - - - -
Common-Name:                  - - - - - - - - - - - - - - -
Options:
Logon to the Default DSA
```

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 1b displays the following fields:

Directory ID

Displays the directory ID of the directory to be administered (entered in Mask 1a).

Authentication mechanism to be used

Displays the authentication method (entered in Mask 1a).

Password Enter your password (password of the administrator object). This field is present only if the authentication method is **Simple Unprotected**.

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. The **Country_name**, **Organization-Name**, **Org.-Unit-Name**, and **Common-Name** fields are only present if the authentication mechanism is **Simple Unprotected**.

If you chose DCE authentication, you do not have to enter any part of the DN of the administrator object. During the **bind** operation, the DN (extended **rgy** attribute **X500_DN**) will be used as it is assigned to the principal that invokes the **gdsditadm** process. The administrator must have performed a DCE login before this.

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 logon 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 onto 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 all **PUBLIC** attributes and attributes that are not protected by an ACL. This implies that you can only get access rights to attributes that are protected by an ACL if you use **SIMPLE** or **DCE** authentication. The **gdsditadm** processs performs a bind with credentials that the target DSA will check.

7.2 Mask 2: DSA Identification

To connect to a specific DSA (by using the **Logon to a Specific DSA** option), the administrator needs to enter the DN in Mask 2 (Figure 7-3).

Figure 7–3. Mask 2: DSA Identification

```
(Mask 2)                              DIRECTORY SYSTEM                    Logon to a Specific DSA


DSA IDENTIFICATION:
Country-Name:                - -
Organization-Name:           - - - - - - - - - - - - -
Org.-Unit-Name:              - - - - - - - - - - - - -
Common-Name:                 - - - - - - - - - - - - -
Common-Name:                 - - - - - - - - - - - - -
Options: None
```

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 (Figure 7-3) 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 7-4). This mask is used to select the required administration functions.

Figure 7–4. 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 by 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 7-5.

Figure 7–5. 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 Using gdscp to Logon to a DSA or the DUA Cache

You can log into a specific or default DSA by using the **gdscp bind** operation. For example, the following **gdscp** command makes a bind with user credentials, to the specified DSA for directory-id **2**:

```
%
bind -dirid 2 -dsa /C=de/O=sni/OU=ap11/CN=dsa15 \
-user /C=de/O=siemens/OU=dap11/CN=miller -password xxx \
-authentication SIMPLE
```

You can also log into the DUA cache. For example, the following **gdscp bind** operation logs onto the DUA cache for directory-id **2**:

```
% bind -cache -dirid 2
```

However, before trying to perform any other operations on the cache, you must set the service controls appropriately. Refer to the *DCE 1.2.2 Application Development Guide* for detailed descriptions of the service controls.

Refer to the *DCE 1.2.2 Command Reference* for detailed information on how to use **gdscp bind** operation.

7.6 XDS API Function Calls and the DUA Cache

Application programs can make directory service calls by 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.

This chapter also describes how to use **gdscp** commands that are equivalent to mask sequences for individual operations.

8.1 Masks

Each of the following sections describes one of the masks used for object administration. See Section 4.1 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 perform 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 nonGDS 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

```
(Mask 4a)                         DIRECTORY SYSTEM                  Display Special DSAs



Special DSAs
Local:    ..................................................
..................................................
Default:  ..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
```

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

```
(Mask 6)                              DIRECTORY SYSTEM                                operation



name       ........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
........................    - - - - - - - - - - - - -
Structural Object Class: object class
       Auxiliary Object Class:
NO
```

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

Refer to 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 enters 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

```
(Mask 6a)                          DIRECTORY SYSTEM                    Modify Attribute


Access Rights
         Modify Public:
NO
         Read Standard:
NO
         Modify Standard:
NO
         Read Sensitive:
NO
         Modify Sensitive:
NO
```

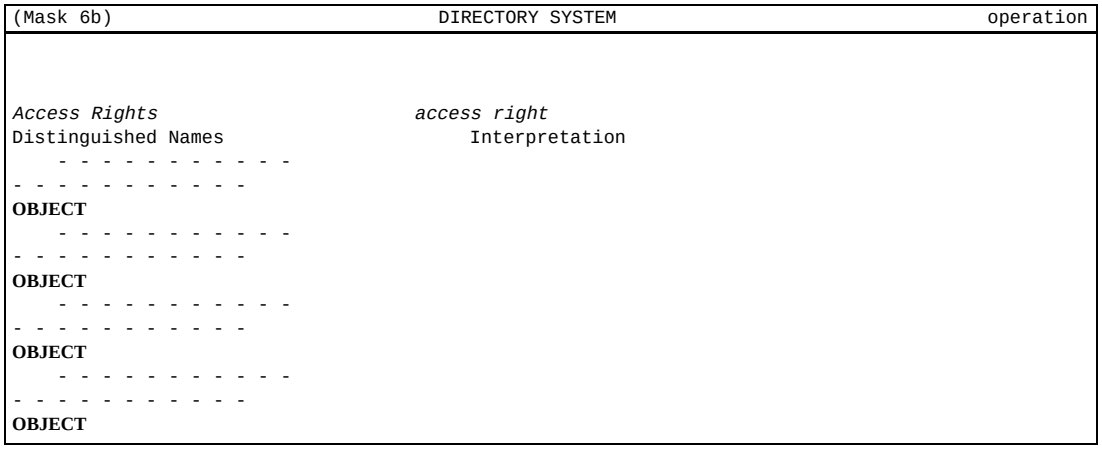
In Mask 6a, the *operation* field can only contain **Modify Attribute**.

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

Refer to 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 by 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)

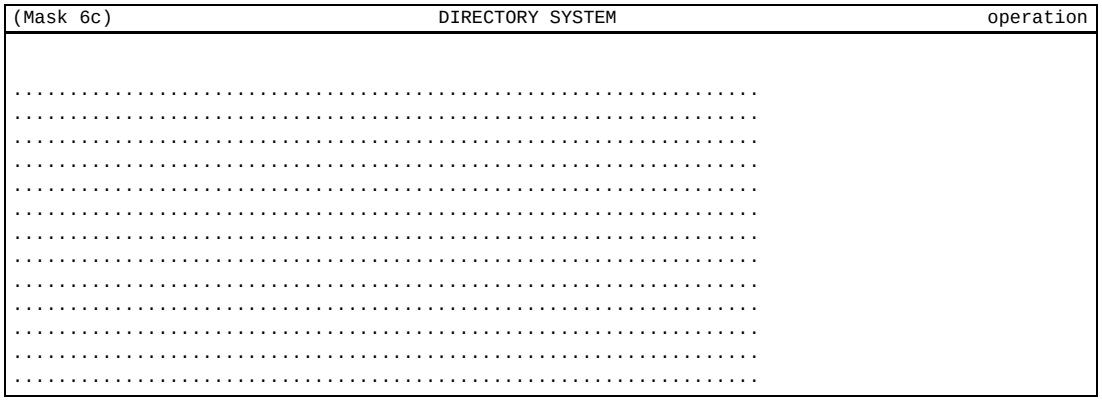
If the access rights are changed, the DNs according to the old ACL are displayed.

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.

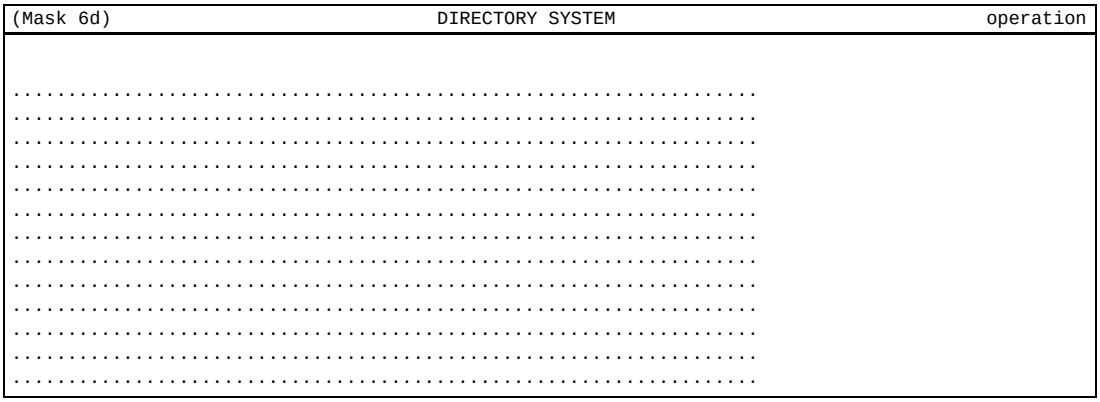
If you do not select any auxiliary object classes, it is assumed that all auxiliary object classes have been selected.

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

Refer to 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)
- DME NMO Alternate Address syntax (Mask 39)

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)
- Certification syntax (Mask 36)
- Certification Pair syntax (Mask 37)
- Certification List syntax (Mask 38a)
- Revoked Certificate List syntax (Mask 38b)

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

```
(Mask 7a)                          DIRECTORY SYSTEM                          operation


P-Selector:                    - - - - - - - - - - - -
S-Selector:                    - - - - - - - - - - - -
T-Selector:                    - - - - - - - - - - - -
NSAP-Address 1:                    - - - - - - - - - - - - - -
NSAP-Address 2:                    - - - - - - - - - - - - - -
NSAP-Address 3:                    - - - - - - - - - - - - - -
NSAP-Address 4:                    - - - - - - - - - - - - - -
NSAP-Address 5:                    - - - - - - - - - - - - - -
```

In Mask 7a, *operation* will be one of the following, depending on the operation being performed:

Add Object
Display Objects
Add Attributes
Modify Attribute

Refer to 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

```
(Mask 21)                          DIRECTORY SYSTEM                          operation


CDS-Cell      Namespace UUID: - - - - - - - - - - - - - -
      Root Dir UUID:  - - - - - - - - - - - - - -
      Root Dir Name:  - - - - - - - - - - - - - -
      Modification: Modify Value
```

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

Refer to 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 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 hexadecimal 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

```
(Mask 22)                              DIRECTORY SYSTEM                                operation


CDS-Replica
Replica Type:       MASTER      Clearinghouse UUID: - - - - - - - - - - - -
      Clearinghouse Name: - - - - - - - - - - - - -
      Tower 1:            - - - - - - - - - - - - -
Tower 2:            - - - - - - - - - - - - -
Tower 3:            - - - - - - - - - - - - -
Tower 4:            - - - - - - - - - - - - -
Tower 5:            - - - - - - - - - - - - -
      Modification:
Modify Value
```

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

Refer to 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 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 hexadecimal 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 a 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 with TTX=ID Syntax

```
(Mask 23)                          DIRECTORY SYSTEM                          operation


attribute name
    Terminal:     - - - - - - - - - -
    Control:      - - - - - - - - - - - -
    Graphic:      - - - - - - - - - - - -
    Miscel:       - - - - - - - - - - - -
    Page Formats: - - - - - - - - - - - -
    Private Use:  - - - - - - - - - - - -
    Modification:
Modify Value
```

In Mask 23, *operation* will be one of the following, depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute

Refer to 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

Refer to 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 with Postal Address Syntax

```
(Mask 25)                          DIRECTORY SYSTEM                          operation


attribute name
        Part 1: - - - - - - - - - - - - - - -
        Part 2: - - - - - - - - - - - - - - -
        Part 3: - - - - - - - - - - - - - - -
        Part 4: - - - - - - - - - - - - - - -
        Part 5: - - - - - - - - - - - - - - -
        Part 6: - - - - - - - - - - - - - - -
      Modification:
Modify Value
```

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

Refer to 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 with Fax Number Syntax

```
(Mask 26)                          DIRECTORY SYSTEM                          operation


attribute name
      Telephone Number: - - - - - - - - - - - - - - - - - -
      A3 Width:
NO
      B4 Length:
NO
      B4 Width:
NO
      Fine Resolution:
NO
      Two Dimensional:
NO
      Uncompressed:
NO
      Unlimited Length:
NO
      Modification:
Modify Value
```

In Mask 26, the *operation* field can contain any of the following:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

Refer to 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.

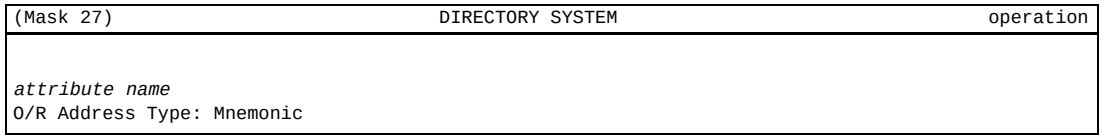
Modification
Enter one of the following values by pressing the space bar:

- **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 Message Handling Systems 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

Refer to 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
Display 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

Refer to 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 (MD) 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

Refer to 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 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 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

Refer to 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* field will be one of the following, depending on the operation being performed:

Add Object
Add Attributes
Display Objects
Modify Attribute
Modify Subtree

Refer to 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 (MD) 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

```
(Mask 33)                          DIRECTORY SYSTEM                          operation


attribute name
DL Submit Permission Type:
Individual
```

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

```
(Mask 34)                          DIRECTORY SYSTEM                          operation


attribute name
DL Submit Permission Type:
type

Directory Name: - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - -
O/R Address:
NO
Modification:
Modify Value
```

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

Refer to 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

Refer to 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:

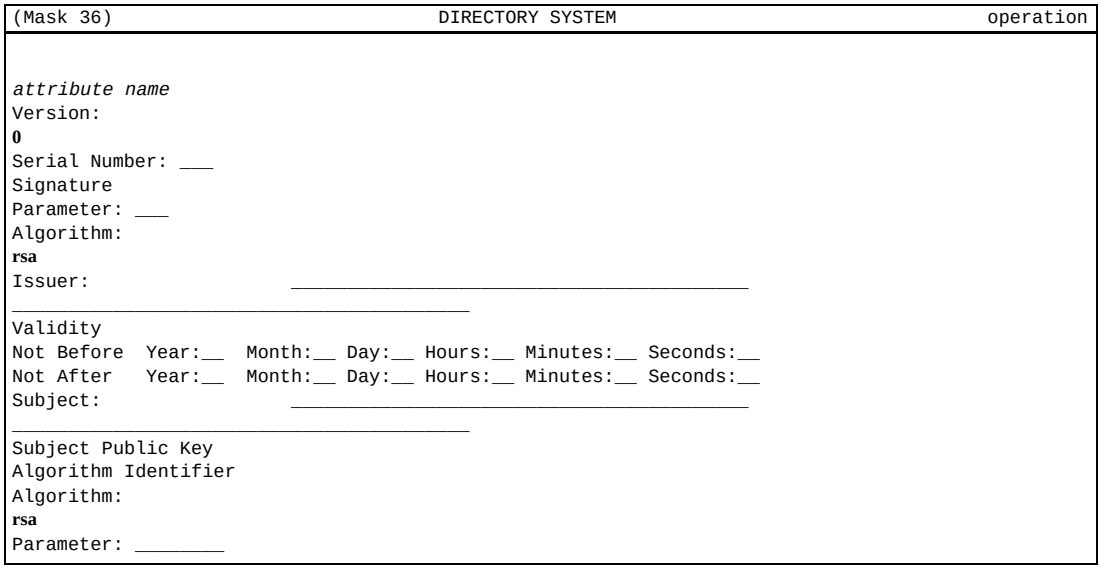
- **Modify Value** (only if *operation* is **Modify Attribute**)
- **Add Value**
- **Delete Value**

8.1.26 Mask 36: Attribute with Certificate Syntax

Mask 36 (Figure 8-28) is used to display an attribute with Certificate syntax. The **User-Certificate** and **CA-Certificate** attribute types use Certificate syntax. These attribute types are used in authentication services by XDS application programs. Authentication services may be offered by the Directory to its users by using the Strong Authentication Package.

You cannot add or modify this security attribute through the GDS administration program. You can only display the values of these attributes. The attribute is added and modified by an XDS application program.

Figure 8–28. Mask 36: Attribute with Certificate Syntax



In Mask 36, the *operation* is **Display Objects**.

Refer to Section 8.2.3 for a description of the **Display Objects** operation.

Mask 36 displays the following fields:

attribute name
Displays the name of the attribute in this field. For example, the attribute name might be **User-Certificate** or **CA-Certificate**.

Version Displays the version number of the certificate.

Serial Number
Displays the serial number of a certificate issued by a certification authority.

Signature Displays the algorithm and parameter used by the signature to sign the certificate. Valid algorithm values are

- **rsa**
- **sqMod-n** (square Modulo-n)

- **sqMod-nWithRSA**

Issuer Displays the DN of the certification authority that issued the certificate.

Validity Displays the time period during which the certificate is valid.

Subject Displays the DN of the user who has been assigned the certificate by the certification authority.

Subject Public Key
Displays the algorithm identifier algorithm and parameter used for the public key. Valid algorithm values are

- **rsa**
- **sqMod-n** (square Modulo-n)
- **sqMod-nWithRSA**

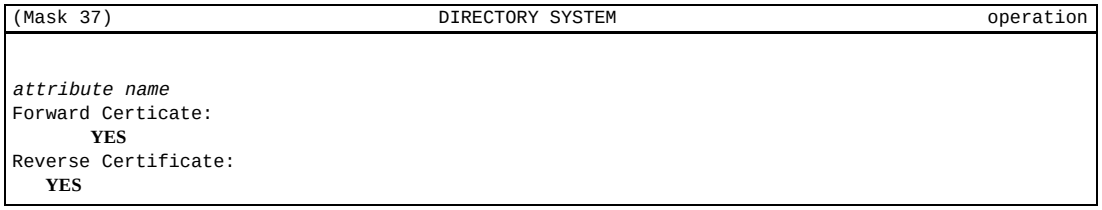
8.1.27 Mask 37: Attribute with Certificate Pair Syntax

Mask 37 (Figure 8-29) is used to display an attribute with Certificate Pair syntax. The **Cross-Certificate-Pair** attribute uses Certificate Pair syntax. This mask is displayed if a forward or reverse certificate is present with Certificate Pair syntax. Certificate pairs are used by certification authorities to cross-certify one another as result of a bilateral agreement. Certification pairs shorten the certification path between the two authorities by reducing the information that must be obtained from the Directory for a particular instance of authentication.

This attribute is used in authentication services by XDS application programs. Authentication services may be offered by the Directory to its users by using the Strong Authentication Package.

You cannot add or modify this security attribute through the GDS administration program. You can only display the values of these attributes. The attribute is added and modified by an XDS application program.

Figure 8–29. Mask 37: Attribute with Certificate Pair Syntax



In Mask 37, *operation* is **Display Objects**.

Refer to Section 8.2.3 for a description of the **Display Objects** operation.

Mask 37 displays the following fields:

attribute name
Displays the name of the attribute; for example, **Cross-Certificate-Pair**.

Forward Certificate
Displays **YES** if there is a forward certificate present (Mask 36 is displayed) and **NO** if no forward certificate is present.

Reverse Certificate
Displays **YES** if there is a reverse certificate present (Mask 36 is displayed) and **NO** if no reverse certificate is present.

8.1.28 Mask 38a: Attribute with Certificate List Syntax

Mask 38a (Figure 8-30) is used to display an attribute with Certificate List syntax. The **Authority-Revocation-List** and **Certification-Revocation-List** attributes use Certification List syntax. The certificate list contains certificates issued by a certification authority. This attribute is used in authentication services by XDS application programs. Authentication services may be offered by the Directory to its users by using the Strong Authentication Package.

You cannot add or modify this security attribute through the GDS administration program. You can only display the values of these attributes. The attribute is added and modified by an XDS application program.

Figure 8–30. Mask 38a: Attribute with Certificate List Syntax

```
(Mask 38a)                          DIRECTORY SYSTEM                          operation


attribute name
Signature                    Algorithm:   _______
Parameter:   _______
Issuer:                   ____________________________________________
_____________________________________
Last Update  Year: __ Month: __ Day: __ Hours: __ Minutes: __ Seconds: __
Revoked Certificates:
  NO
```

In Mask 38a, *operation* will be **Display Objects**

Refer to Section 8.2.3 for a description of the **Display Objects** operation.

Mask 38a displays the following fields:

attribute name
: Displays the name of the attribute; for example, **Authority-Revocation-List** or **Certificate-Revocation-List**.

Signature
: Displays the algorithm and parameter used by the signature to sign the certificate.

Issuer
: Displays the DN of the certification authority that issued the certificate.

Last Update
: Displays the date and time of the last update of the certificate by the certification authority.

Revoked Certificates
: Displays **Yes** if there are revoked certificates present. Press the **<Return>** or the **<Menu>** key to display Mask 38b (Revoked Certificate component) to view revoked certificates. If the **Revoked Certificates** field displays **NO**, there are no revoked certificates present.

8.1.29 Mask 38b: Attribute with Certificate List Syntax (Revoked Certificates)

Mask 38b (Figure 8-31) displays an attribute with Certificate List syntax (Revoked Certificate Component). Each revoked certificate in the certificate list is displayed in a separate mask and is referred to as a Revoked Certificate Component. Mask 38b is displayed when the **Revoked Certificates** field in Mask 38a contains **YES** and you press the **<Return>** or the **<Menu>** key.

As a component of an attribute with Certificate List syntax, a Revoked Certificate Component is used in authentication services by XDS application programs. Authentication services may be offered by the Directory to its users by using the Strong Authentication Package.

You cannot add or modify this security attribute through the GDS administration program. You can only display the values of these attributes. The attribute is added and modified by an XDS application program.

Figure 8–31. Mask 38b: Attribute with Revoked Certificate List Syntax

```
(Mask 38b)                          DIRECTORY SYSTEM                          operation


attribute name "
Revoked Certificate
Signature                    Algorithm:   _______
Parameter:   _______
Issuer:                   ____________________________________________
_____________________________________
User Certificate:         __________
Revocation Date
Year: __ Month: __ Day: __ Hours: __ Minutes: __ Seconds: __
Other Revoked Certificates:
  NO
```

In Mask 38b, *operation* is **Display Objects**.

Refer to Section 8.2.3 for a description of the **Display Objects** operation.

Mask 38b displays the following fields:

attribute name
Displays the name of the attribute: for example, **Authority-Revocation-List**.

Signature Displays the algorithm and parameter used by the signature to sign the certificate.

Issuer Displays the DN of the certification authority that issued the certificate.

Serial Number
Displays the serial number of the certificate.

Revocation Date
Dispays the date and time of the last update of the certificate by the certification authority.

Other Revoked Certificates
Displays **Yes** if other revoked certificates are present. Use the **<Return>** or the **<Menu>** key to view more revoked certificates in the list. This field displays **NO** if there are no more revoked certificates present.

8.1.30 Mask 39: DME NMO Alternate Address

Mask 39 (Figure 8-32) is used to display and add attribute values for a **DME NMO Alternative Address** attribute.

This attribute is used by DME NMO for storing addresses of CMIP objects that are accessed through protocols that are not pure OSI. This attribute is mandatory for **Dme-Nmo-Agent** objects. This multivalued attribute contains an *address* field for storing these alternative addresses in octet string format, followed by 10 fields that can be used to identify the alternative protocols, which are identified by object identifiers.

Figure 8–32. Mask 39: DME NMO Alternative Address

```
(Mask 39)                              DIRECTORY SYSTEM                              operation


DME NMO Alternate Address
Address:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Protocol:
Object ID. 1:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 2:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 3:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 4:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 5:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 6:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 7:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 8:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID. 9:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Object ID.10:  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

In Mask 39, *operation* will be one of the following, depending on the operation being performed:

Add Object
Display Objects

Refer to Section 8.2.3 for a description of the **Display Objects** operation.

Mask 39 displays the following fields:

Address Enter the DME NMO address in Octet String syntax.

Object Id.1 to **Object Id.10**
Enter the object identifier in Object Identifier syntax.

8.1.31 Mask 8: Attribute (Modify)

Mask 8 (Figure 8-33) is used to modify an attribute value that does not require a special mask.

Figure 8–33. Mask 8: Attribute (Modify)

```
(Mask 8)                          DIRECTORY SYSTEM                    Modify Attribute



Attribute:
Name:
attribute name

Old Value:  - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
New Value:  - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - -
```

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.

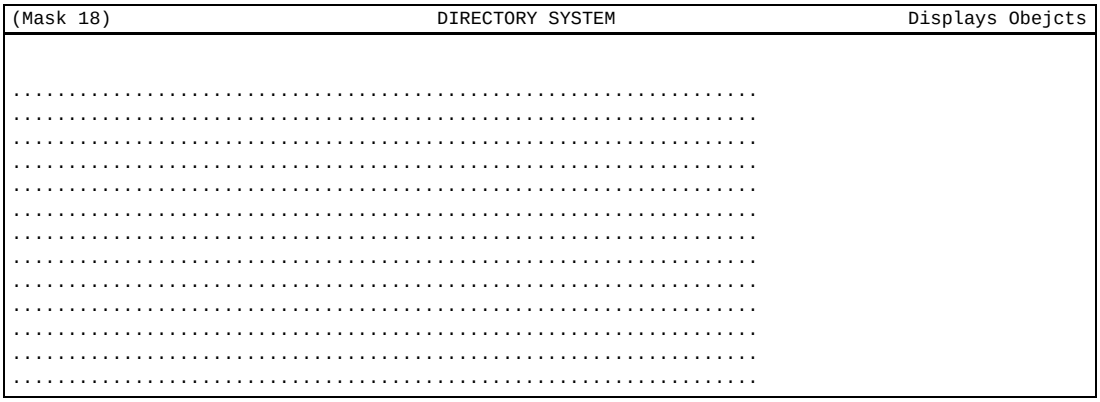
The last attribute value of a recurring attribute can only be deleted with the **Delete Attributes** function.

The first attribute value of a recurring attribute can only be added with the **Add Attributes** function.

8.1.32 Mask 18: Object List

Mask 18 (Figure 8-34) is used to display the results of display operations that cover more than one object.

Figure 8–34. 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. The sections include descriptions of how to use the menu-driven adminstration interface and equivalent **gdscp** commands.

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 for 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.

8.2.1.1 Mask Sequence

Use the following mask sequence for this command:

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)
- **DME NMO Alternate Address** attribute (Mask 39)
- Telex Terminal 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)

  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)

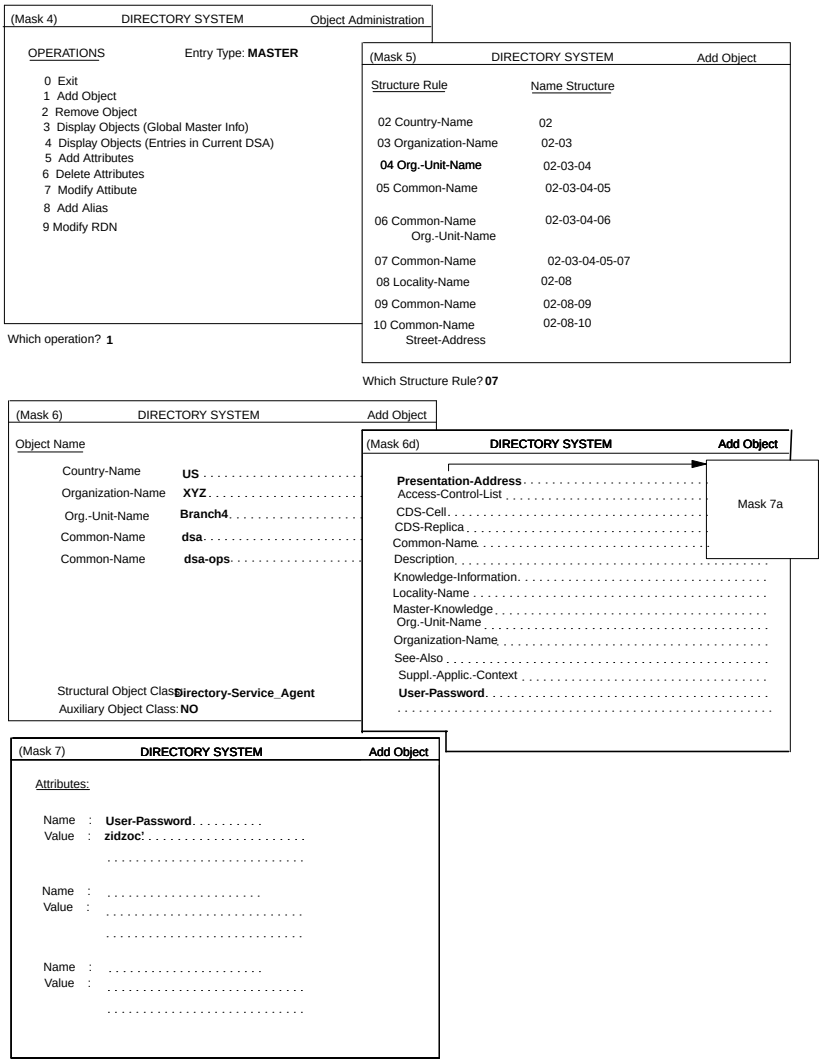
  — Terminal O/R Address (Mask 32)
- MHS DL Submit Permission syntax (Mask 33)

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 8-35 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 DN of the DSA object is **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops**.

Figure 8–35. 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, by 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 by 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 8-35 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 8-35 does not include any special masks.)

8.2.1.2 Adding Objects by Using gdscp

You can use the **gdscp create** operation to add an object to the DIT.

The following example uses command-line mode and creates the object **/C=de/O=sni/OU=ap11/CN=mueller** with the specified attributes:

```
%
gdscp -c create "bind;create /C=de/O=sni/OU=ap11/CN=mueller \
-attribute "OCL=ORP;PER
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object adminstration.

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 for 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.

8.2.2.1 Mask Sequence

Use the following mask sequence for this command:

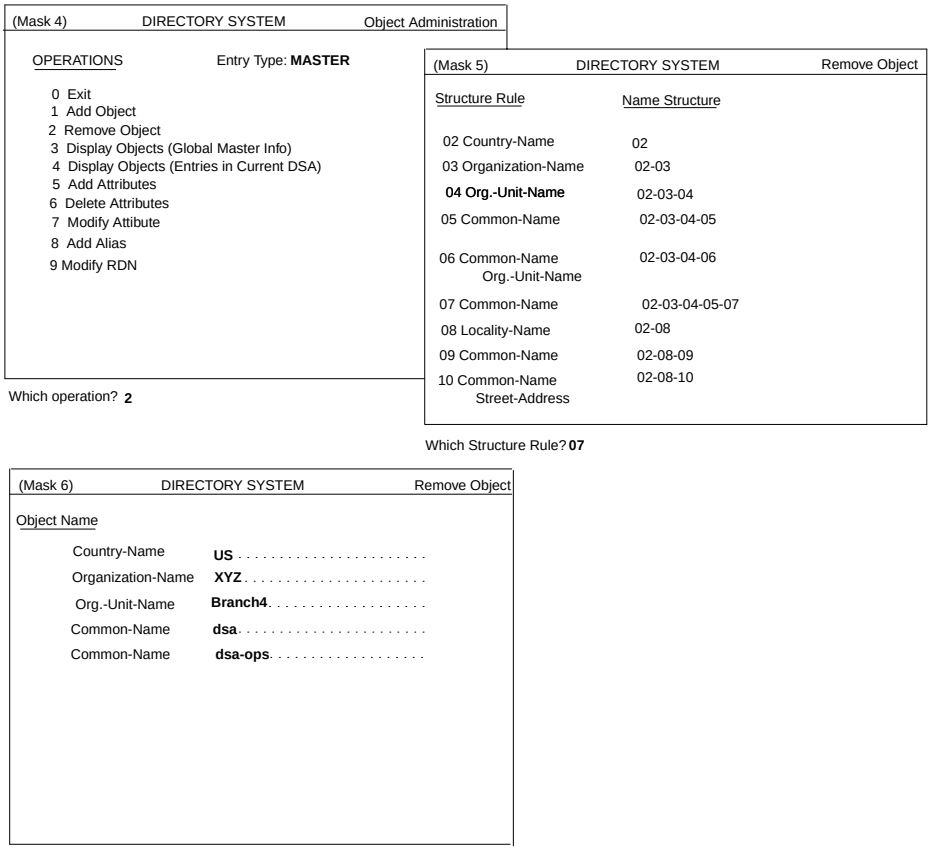
Mask 4 Select option number 2.

Mask 5 Enter the structure rule of object to be deleted.

Mask 6 Enter the DN of object to be deleted.

Figure 8-36 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 8–36. 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>**.

8.2.2.2 Removing Objects by Using gdscp

You can use the **gdscp delete** operation to remove an object from the DIT.

The following example uses command line mode to delete the object **/C=de/O=sni/OU=ap11/CN=mueller** from the dit:

```
%
gdscp -c "bind;delete /C=de/O=de/OU=ap11/CN=mueller"
```

Refer to the **gdscp** and **x500obj** reference page for a complete description of how to use **gdscp** for object adminstration.

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.

8.2.3.1 Mask Sequence

Use the following mask sequence for this command:

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).

Mask 36 Displays the attribute with Certificate syntax.

Mask 37, 38 Displays the attribute with Certificate Pair syntax.

Mask 38a, 38b
Displays the attribute with Certificate List syntax.

Mask 39 Displays the attibute with the DME NMO Alternate Address.

You can page up and down in Masks 6, 6b, 7, 7a, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38a, and 39 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, 35, 36, 37 ,38a, 38b, and 39:

<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.

8.2.3.2 Displaying Objects by Using gdscp

You can use the **gdscp list**, **show**, and **search** operations to display objects from the DIT.

The following example uses command-line mode to list the children of the specified object in the DIT.

```
%
gdscp -c "bind;list /C=de/O=sni -pretty"
```

The returned list is displayed in structured format as follows:

```
1) /C=de/O=sni/OU=ap11/CN=mueller
2) /C=de/O=sni/OU=ap11/CN=schmid
```

The following example shows the output of a **search** command that searches the subtree for all attributes of the object **/C=de/O=siemens/OU=ap11** with the *Common-Name***Brown** or **Andrews**. The example also demonstrates the formatting produced by the **-pretty** option.

```
%
search C=de/O=siemens/OU=ap11 -pretty -subtree -filter \
"CN=Brown | CN=Andrews" -allattr
```

```
 1) Object: /C=de/O=sni/OU=ap11/CN=Brown
Object-Class              : MHS-Distribution-List
```

```
: Top
Common-Name               : Brown
MHS-DL-Submit-Permission
Permission-Type         : 0
Individual
Directory-Name        : /C=de/O=sni
ADMD-Name             : dbp
Common-Name           : Alfred Brown
Country-Name          : de
Domain-Type-1         : MS MAIL
Domain-Value-1        : Brown
Generation            : 396
Given-Name            : Alfred
Initials              : P.
Organization          : Siemens Nixdorf
Organizational-Unit-1 : Munich
Organizational-Unit-2 : P1
Organizational-Unit-3 : P4
Organizational-Unit-4 : ap113
PRMD-Name             : sni
Surname               : Brown
MHS-OR-Address
ADMD-Name               : admd
Country-Name            : de
PRMD-Name               : prmd
Description               : Software Consultant
2) Object: /C=de/O=sni/OU=ap11/CN=Andrews
Object-Class              : MHS-Distribution-List
: Top
Common-Name               : Andrews
MHS-DL-Submit-Permission
Permission-Type         : 0
Individual
Directory-Name        : /C=de/O=sni
ADMD-Name             : dbp
Common-Name           : Peter Andrews
Country-Name          : de
Domain-Type-1         : MS MAIL
Domain-Value-1        : Peter
Generation            : 396
```

```
Given-Name            : Peter
Initials              : P.
Organization          : Siemens Nixdorf
Permission-Type         : 0
Individual
Directory-Name        : /C=de/O=sni
ADMD-Name             : dbp
Common-Name           : Peter Andrews
Country-Name          : de
Domain-Type-1         : MS MAIL
Domain-Value-1        : Peter
Generation            : 396
Given-Name            : Peter
Initials              : P.
Organization          : Siemens Nixdorf
Organizational-Unit-1 : Munich
Organizational-Unit-2 : P1
Organizational-Unit-3 : P4
Organizational-Unit-4 : ap113
PRMD-Name             : sni
Surname               : Andrews
MHS-OR-Address
ADMD-Name               : admd
Country-Name            : de
PRMD-Name               : prmd
Description               : Project Leader
```

The following example reads the current object and displays the values for the **SN** and **BC** attributes:

```
% show -attribute SN BC -pretty
1)  Object:  /C=de/O=sni/OU=ap11/CN=schmid
Surname         : schmid
Business-Category : Software Consultancy
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object adminstration.

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.

The **Add Attribute** operation can only be used to add new attributes to an object. It cannot be used to add new attribute values to an existing recurring attribute. Use the **Modify Attribute** operation to add new attribute values to an existing attribute.

Modify rights (ACLs) are required for 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.

8.2.4.1 Mask Sequence

Use the following mask sequence for this command:

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 syntax (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)

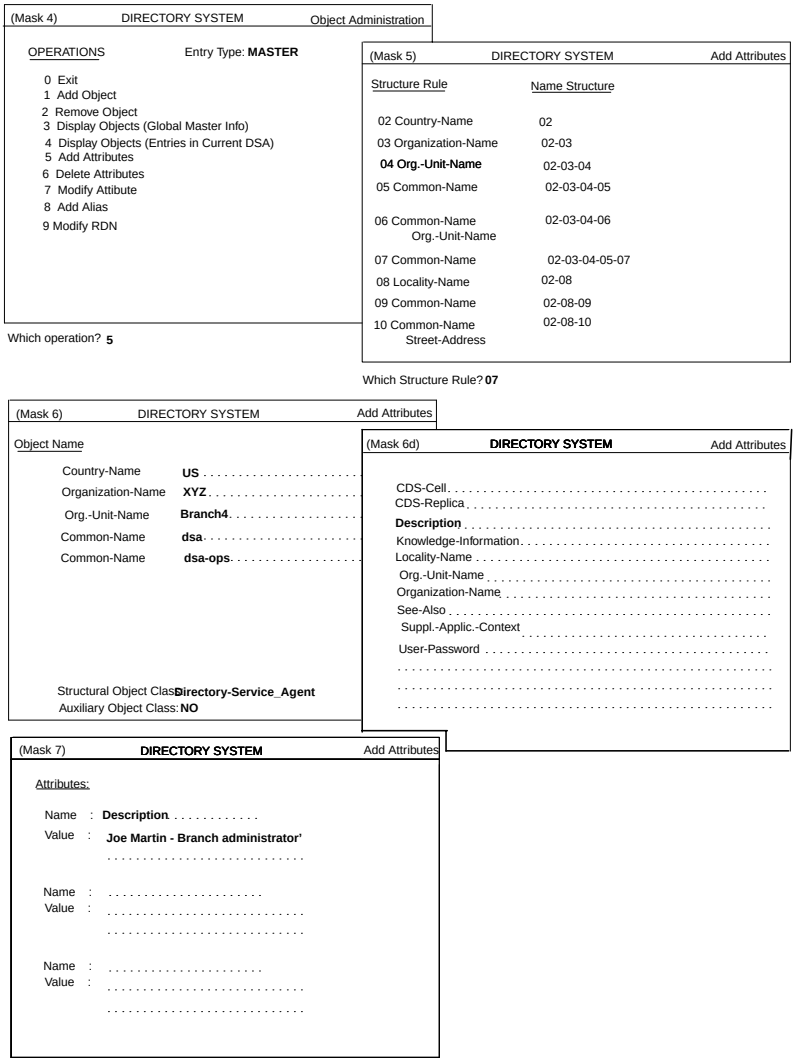
  — Unstructured Postal O/R Address (Mask 31)

  — 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 8-37 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 8–37. 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.

8.2.4.2 Adding Attributes and Attribute Values by Using gdscp

You can use the **gdscp modify** operation to add attributes and attribute values of an object in the DIT.

The following example uses command-line mode and adds the **Description** attribute to the object **/C=de/O=siemens/OU=dap11/CN=mueller** with the values **xxx** and **yyy** if the attribute does not already exist. Otherwise, it adds the new values to the attributes.

```
%
gdscp -c "bind;modify /C=de/O=siemens/OU=dap11/CN=mueller \
-addattr "DSC=xxx;yyy""
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object adminstration.

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**
- **Master-Knowledge**
- **ACL**
- **Aliased-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.

8.2.5.1 Mask Sequence

Use the following mask sequence for this command:

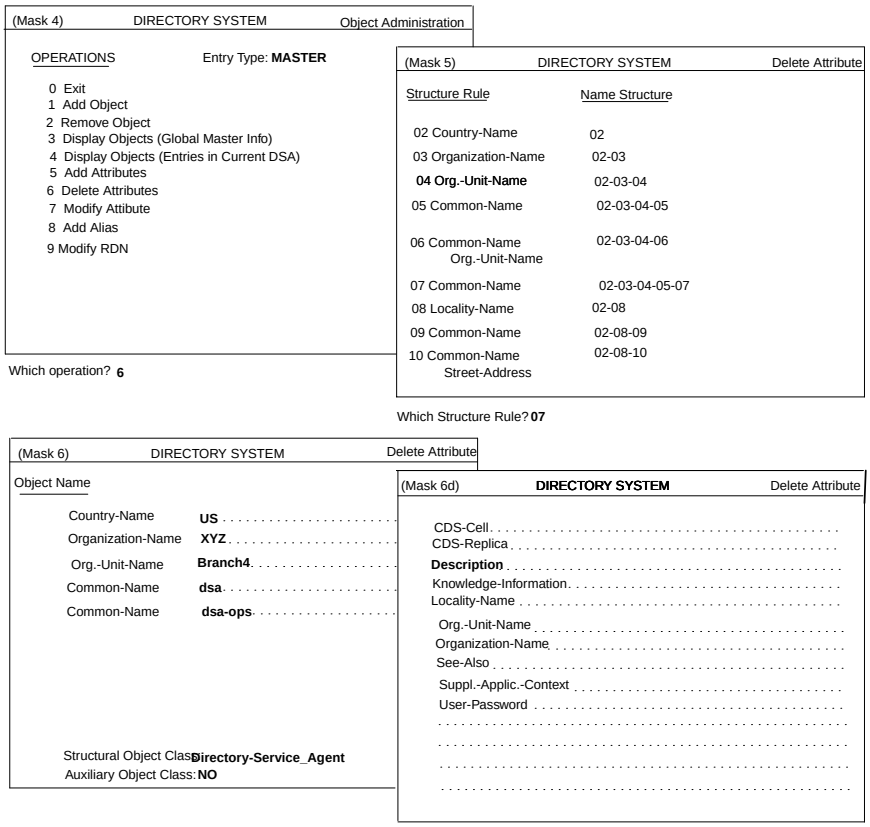
Mask 4 Select option number 6.

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 8-38 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 8–38. 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 by 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.

8.2.5.2 Deleting Attributes and Attribute Values by Using gdscp

You can use the **gdscp modify** operation to delete attributes and attribute values of an object in the DIT.

The following example uses command-line mode and deletes two values of the **DSC** (**Description**) attribute and deletes attributes **L** and **BC** from the specified object entry:

```
%
gdscp -c "bind;modify /CN=de/O=sni/OU=app11/CN=mueller \
-removeattr "DSC=xxx;yyy" L BC"
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object adminstration.

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 for 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.

8.2.6.1 Mask Sequence

Use the following mask sequence for this command:

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 by 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. To add an additional value to a recurring attribute, enter the value in the **New Value** field and make sure that the **Old Value** field is empty.

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.

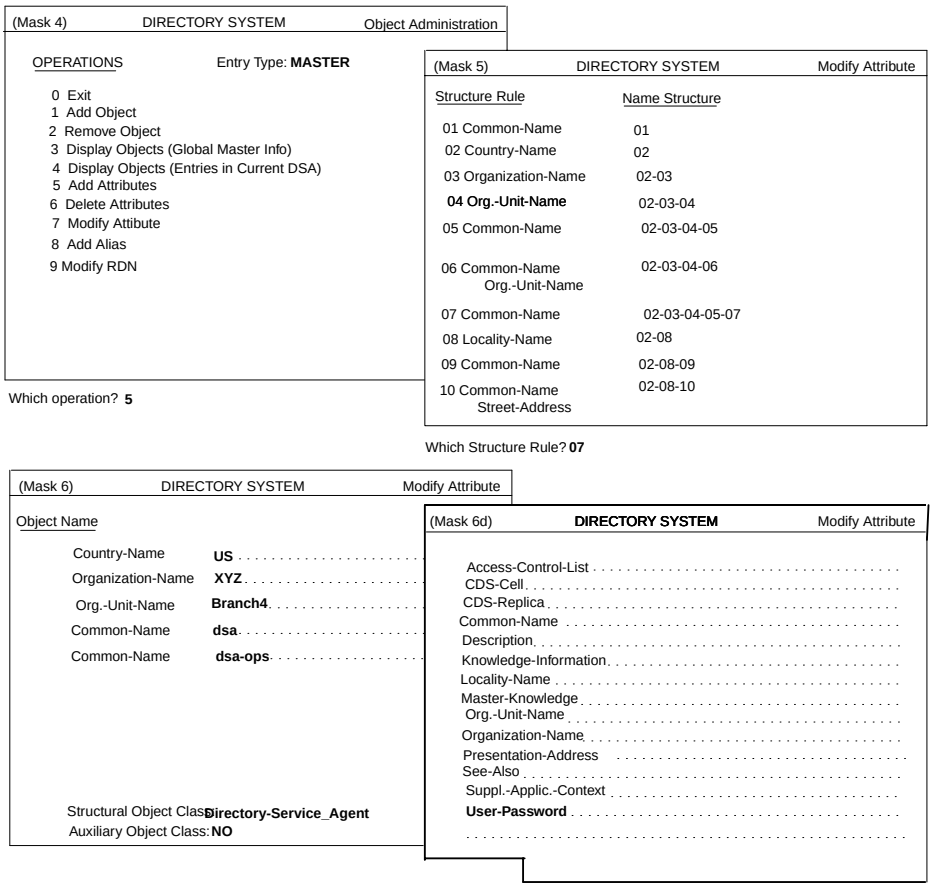
Mask 33 Overwrite attribute value with MHS DL Submit Permission syntax (**Individual**, **Member of DL**, **Pattern Match**) by new value.

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 8-39 shows the masks involved in a sample **Modify Attribute** operation.

Figure 8–39. 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 by 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 by 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 perform the **Modify Attribute** operation.

8.2.6.2 Modifying Attributes and Attribute Values By Using gdscp

You can use the **gdscp modify** operation to modify attributes and attribute values of an object in the DIT.

The following example uses command line mode to replace the **xxx** value of the **SN** attribute with **yyy** and the **aaa** value of the **DSC** attribute with **bbb**:

```
%
gdscp -c "bind;modify /C=ce/O=sni/OU=ap11/CN=mueller -changeattr \
SN=xxx SN=yyy -changeattr DSC=aaa DSC=bbb"
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object adminstration.

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 by 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 4 Select option number 8.

Mask 5 Select the structure rule of the alias to be added.

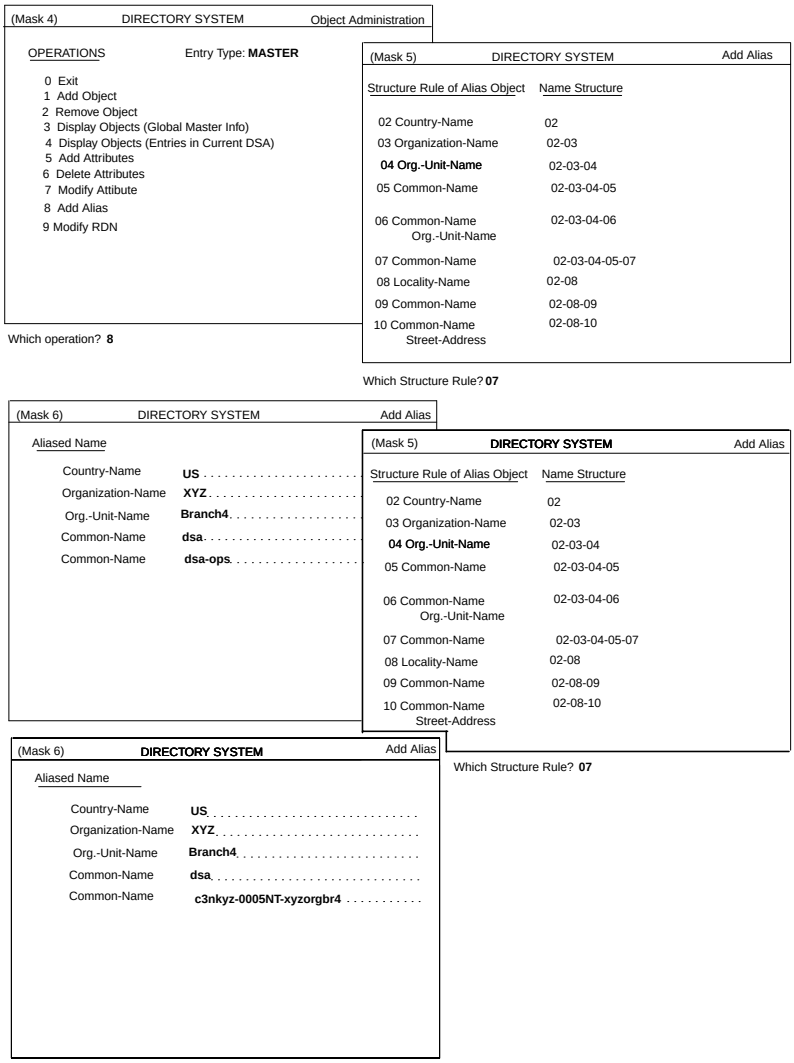
Mask 6 Enter the name of the alias to be added.

Mask 5 Select the structure rule of the aliased object.

Mask 6 Enter the name of the aliased object.

Figure 8-40 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 DIT, where 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 8–40. 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>**.

8.2.7.1 Adding Aliases by Using gdscp

You can use the **gdscp create** operation to create an alias object in the DIT. The following example uses command-line mode to create the alias **/C=de/O=sni**, which is an alias of **/C=de/O=Siemens-Nixdorf**:

```
%
gdscp -c "bind; create C=de/O=sni \
-attribute AON=/C=de/O=Siemens-Nixdorf ocl=ALI"
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object administration.

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 for 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.

8.2.8.1 Mask Sequence

Use the following mask sequence for this command:

Mask 4 Select option number 9.

Mask 5 Enter the structure rule of the object.

Mask 6 Enter the object name of the object whose last RDN has to be changed.

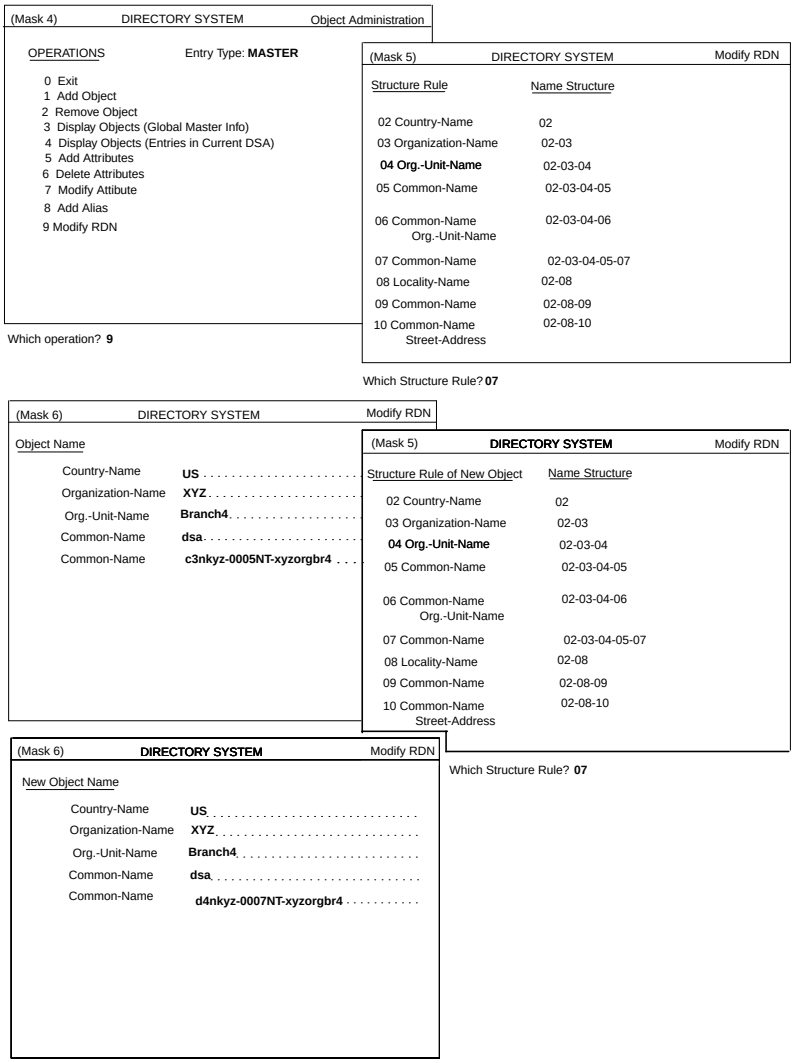
Mask 5 Enter the structure rule of the new object.

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 8-41 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, where 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 8–41. 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**.

8.2.8.2 Modifying the RDN of an Object by Using gdscp

You can use the **gdscp modify** operation to modify the RDN of an object in the DIT.

The following example uses command-line mode to modify the RDN; as a result, **/C=de/O=siemens/OU=dap11/CN=mueller,OU=sni** becomes the new object name:

```
%
gdscp -c "bind;modify /C=de/O=siemens/OU=dap11/CN=mueller \
-rdn CN=mueller,OU=sni"
```

Refer to the **gdscp** and **x500obj** reference pages for a complete description of how to use **gdscp** for object adminstration.

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.

8.2.9.1 Mask Sequence

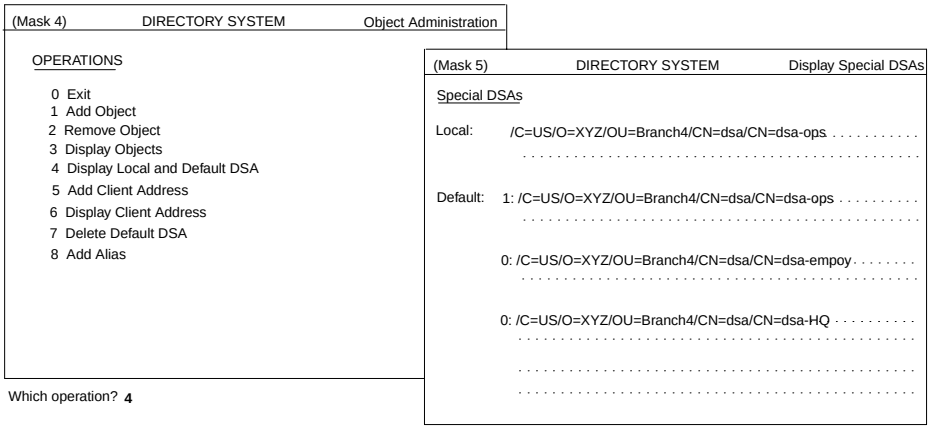
Use the following mask sequence for this command:

Mask 4 Select option number 4.

Mask 4a Displays the DN of local and default DSAs.

Figure 8-42 shows how the DNs of the local and default DSAs stored in the DUA cache are displayed by using Masks 4 and 4a.

Figure 8–42. 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.

8.2.10.1 Mask Sequence

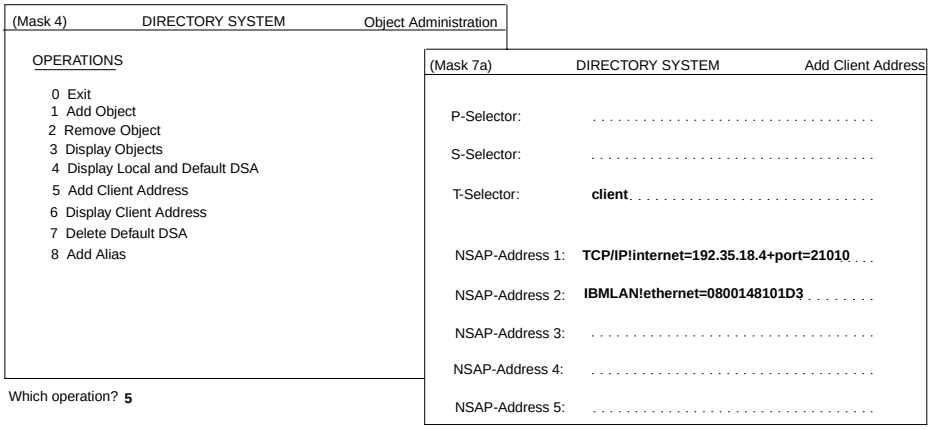
Use the following mask sequence for this command:

Mask 4 Select option number 5.

Mask 7a Enter the presentation address.

Figure 8-43 shows how a client address is added to the DUA cache by using Masks 4 and 7a. Input is highlighted in bold.

Figure 8–43. 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.

8.2.11.1 Mask Sequence

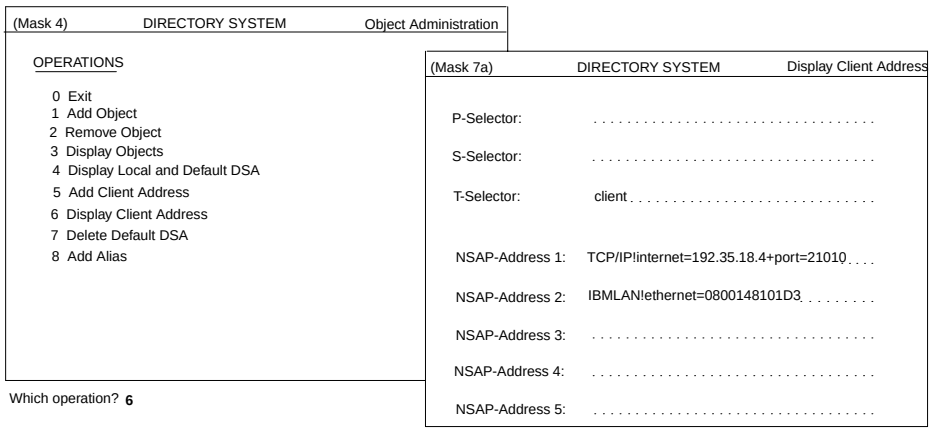
Use the following mask sequence for this command:

Mask 4 Select option number 6.

Mask 7a Displays the presentation address.

Figure 8-44 shows how a client address stored in the DUA cache is displayed by using Masks 4 and 7a.

Figure 8–44. 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.

8.2.12.1 Mask Sequence

Use the following mask sequence for this command:

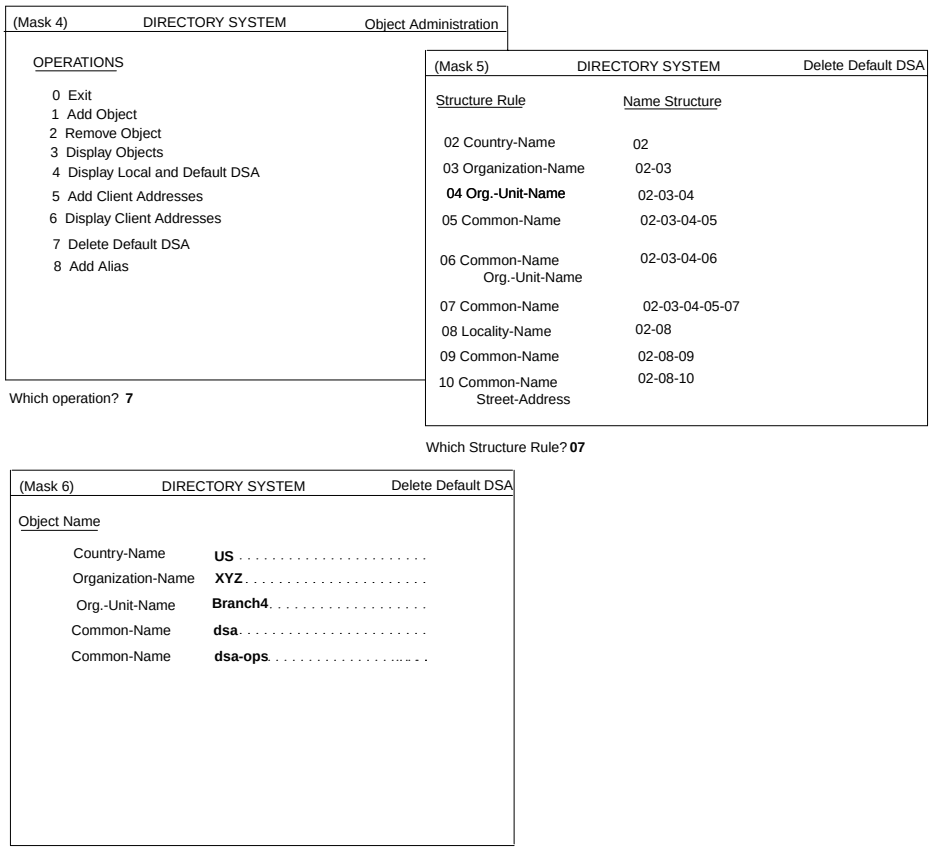
Mask 4 Select option number 7.

Mask 5 Select the structure rule of the default DSA.

Mask 6 Enter the DN of the default DSA to be deleted.

Figure 8-45 shows how to delete a default DSA that is stored in the DUA cache.

Figure 8–45. 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 with 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. (For further information, see Recommendation X.520 and X.411.)

  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 by 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. The 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. The 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	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	Only TRUE and FALSE are permitted (according to X.208 standard); matches for equality.
Integer	Only integers are permitted (according to X.208 standard); matches for equality and ordering.
UTC-Time	Absolute time (by 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	Telephone numbers in Printable String format; matches for equality and substrings (spaces, dashes, slashes are skipped during comparison).
Search Guide	Complex recursive attribute syntax; components are Object Identifier and Criteria; no matching.
Password	Any byte sequence (according to X.208 standard); matches for equality only.
Country Name	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.
Presentation Address	Consists of a Set of NSAP addresses and, optionally, P-Selector, S-Selector and T-Selector; matches for equality.

Syntax	Description
Fax Number	Consists of a telephone number (Printable String) and, optionally, the G3 facsimile nonbasic parameters; matches for equality.
TTX-ID	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 7-4). 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). For further information, see Chapter 10.

Note: A schema specification file, **XOI_SCHEMA_FILE**, is used by **gdscp** to read some of the details related to the attributes and object classes. If you change the schema, you must edit this file so that the attributes displayed by **gdscp** reflect the attributes in the new schema. Refer to the Appendix K for information on the format of the file and how to edit it.

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

```
(Mask 10)                          DIRECTORY SYSTEM                          operation


Rule Number:                       - -
Superior Rule Number:              - -
Acronyms of Naming Attributes:     - - - - - - - - - - -
Structural Object Class:           - - -
```

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.

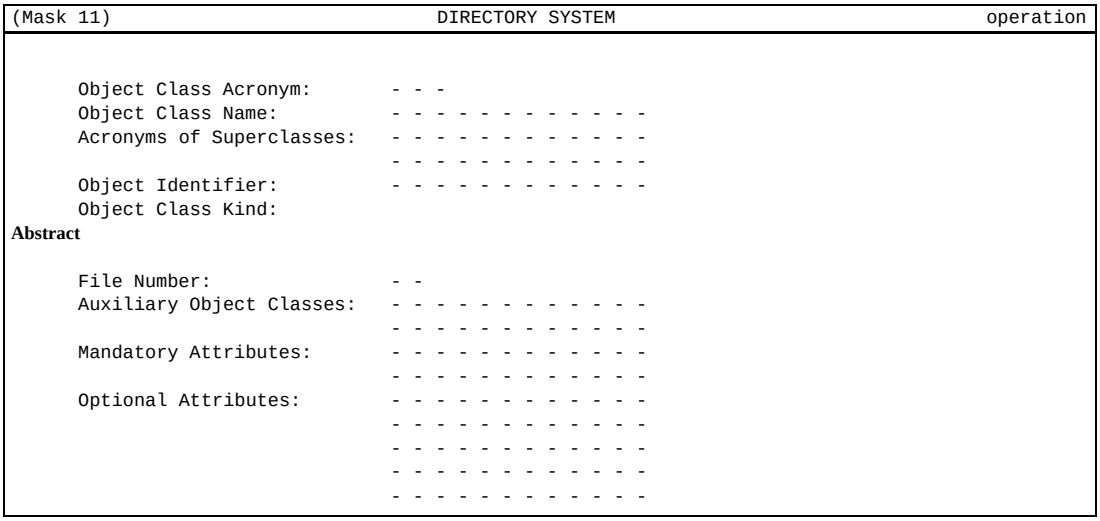
<Scroll Up>
Display the previous SRT element.

<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



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

```
(Mask 12)                              DIRECTORY SYSTEM                                operation


Attribute Acronym:          - - -
Attribute Name:             - - - - - - - - - - - - -
Object Identifier:          - - - - - - - - - - - - -
Lower Bound:                - - - -
Upper Bound:                - - - -
Number of Recurring Values: - - - -
      Attribute Syntax:
Case Ignore String
      Access Class:
Standard
      Index Level:                - -
Phonetic Matching:
NO
```

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

- Search Guide
- Password
- Country Name
- Presentation Address
- Telephone Number
- Fax Number
- TTX-ID syntax
- Telex Number
- Postal Address
- Case Ignore List
- Boolean
- Integer
- UTC Time
- Numeric String
- Printable String
- Object Identifier
- DN
- Preferred Delivery Method
- Any
- ASN.1
- MHS O/R Address
- MHS O/R Name
- MHS DL Submit Permission
- MHS Preferred Delivery Method

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, an attribute could be **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 (in **/opt/dcelocal/var/directory/gds/dsa/dir***x*, where *x* is the directory ID) the **asn1_attr** file 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.

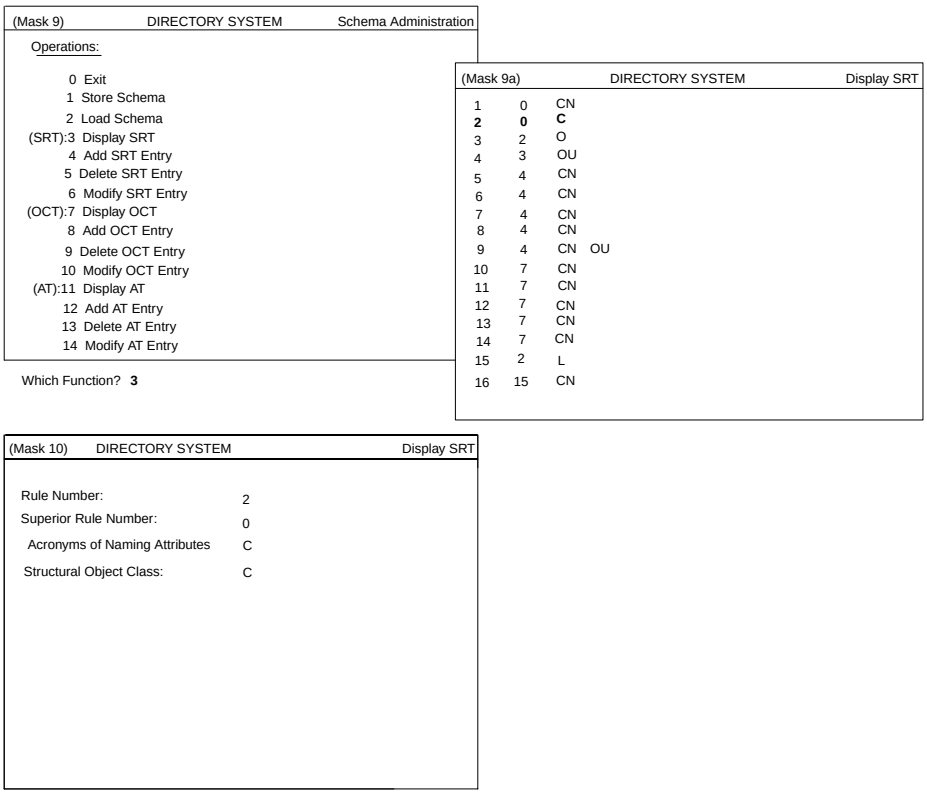
Mask 9a Select the element to be displayed.

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

The **Add SRT Entry** operation adds a new structure rule to the SRT.

The operation adds the new structure rule in 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.

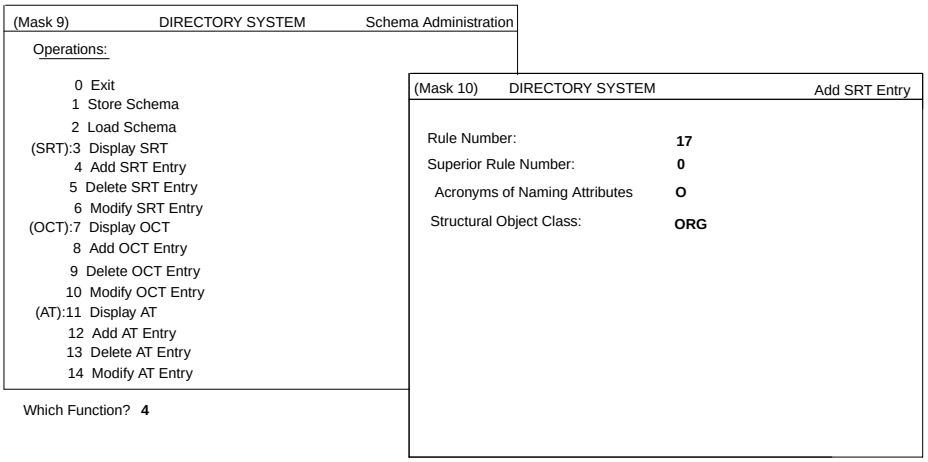
Mask 10 Enter the structure rule.

The following conditions must exist for this operation to complete successfully:

- The Naming Attributes must do the following:
  - — Exist in the AT
  - — Belong to an object class
  - — Cannot be real recurring attributes (that is, with an unlimited number of attribute values)
  - — Have either Case Ignore String, Case Exact String, Printable String, or T61 String syntax.
- 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

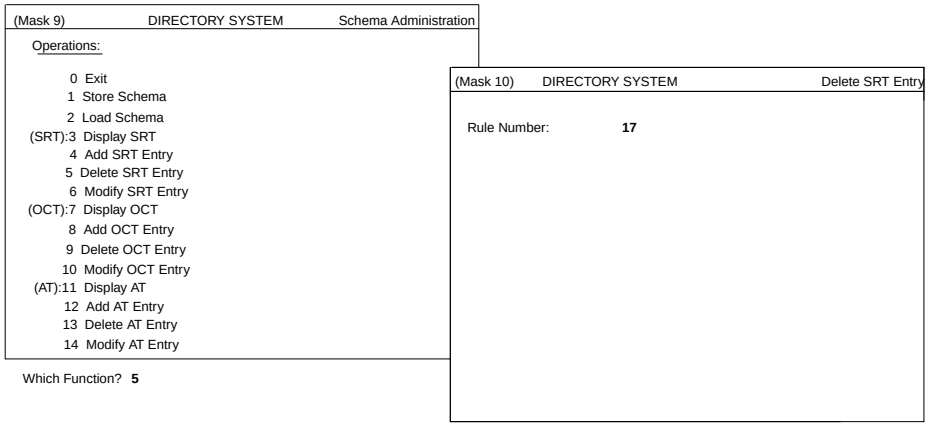
Mask 9 Select option number 5.

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 a 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

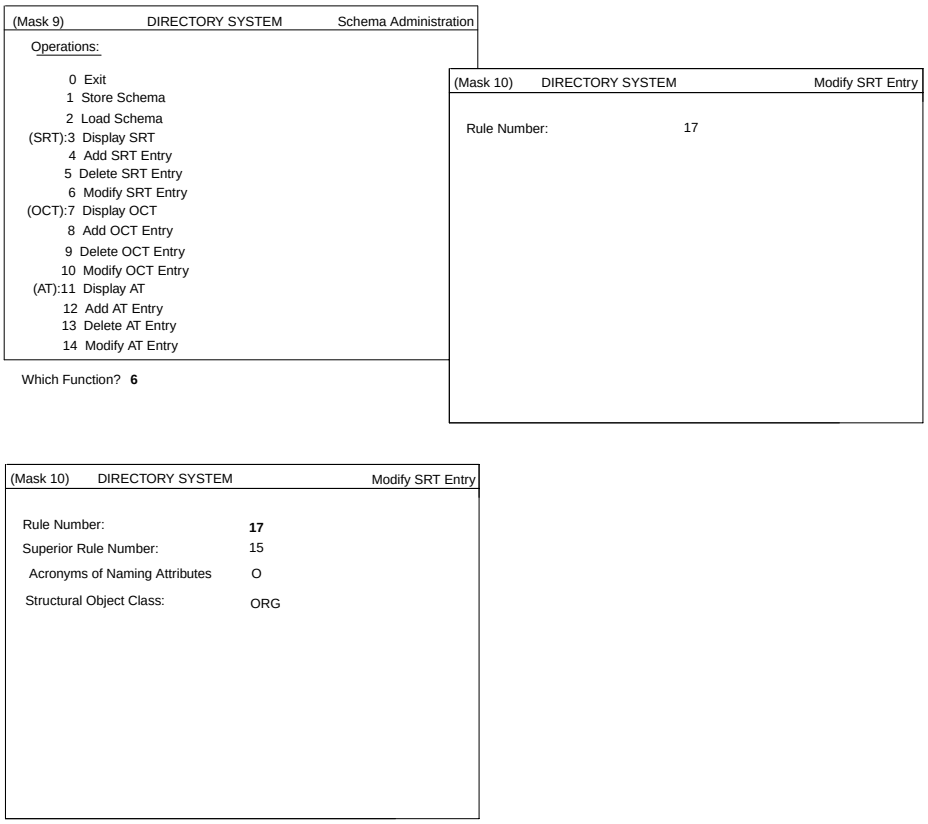
Mask 9 Select option number 6.

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

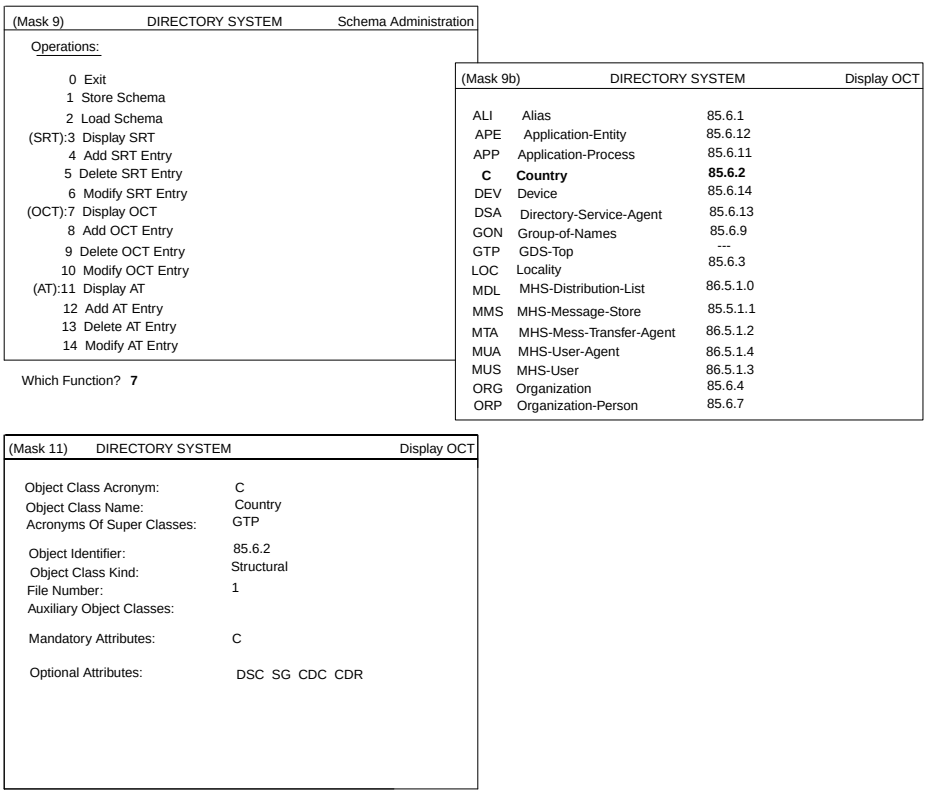
Mask 9 Select option number 7.

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

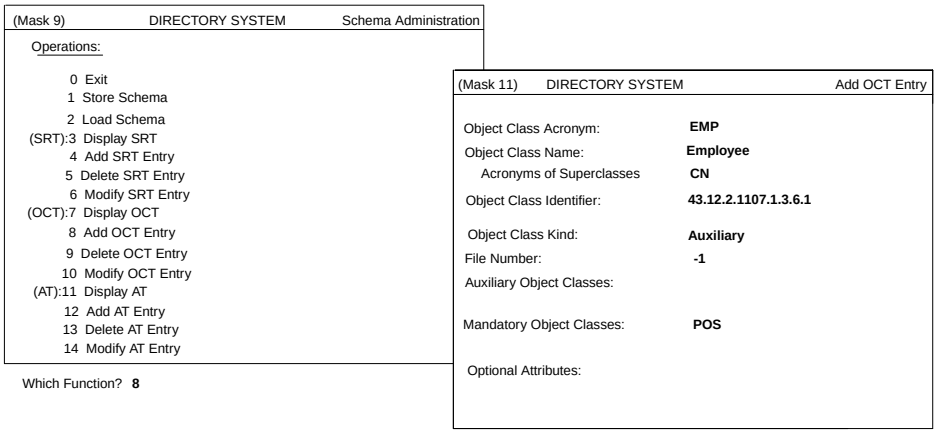
Mask 9 Select option number 8.

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.

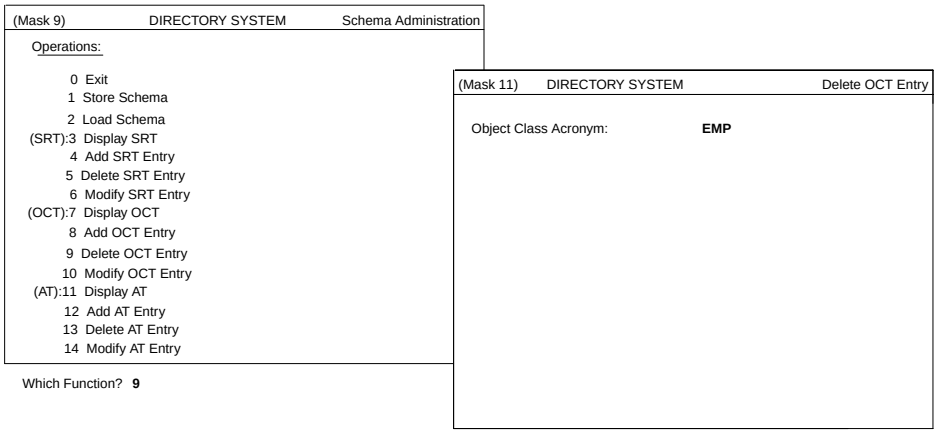
Mask 11 Select the object class to be deleted.

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.

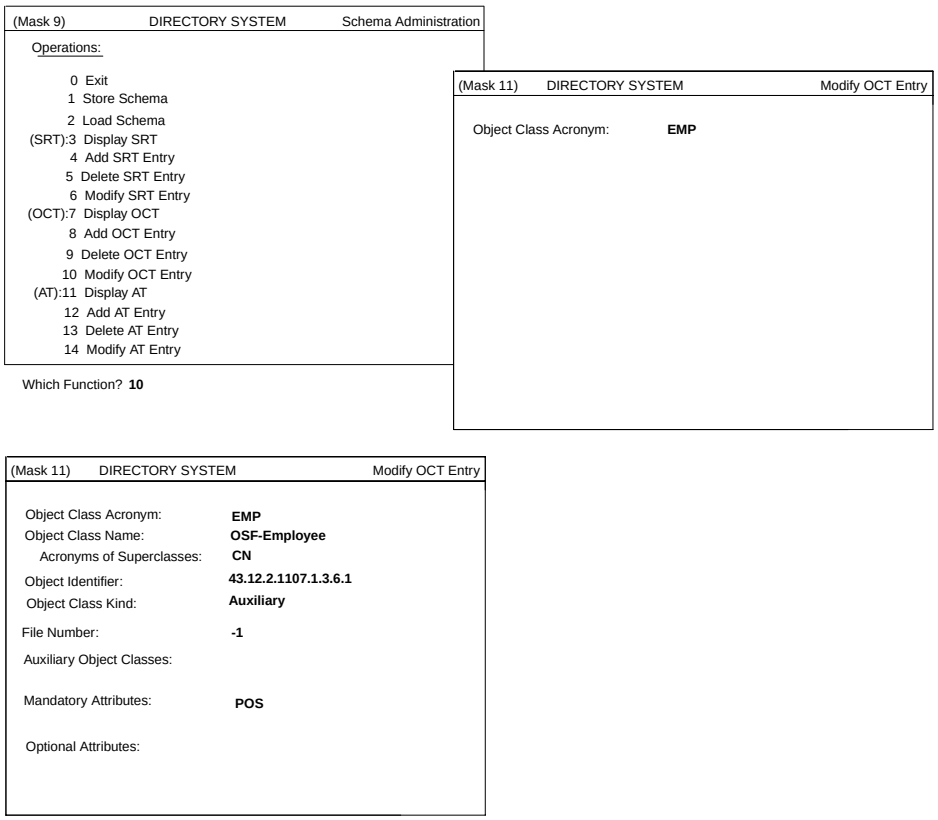
Mask 11 Only the **Object Class Acronym** field is displayed initially. When a correct object class acronym is entered, Mask 12 is redisplayed with all values belonging to the OCT entry. Fields can then be modified.

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

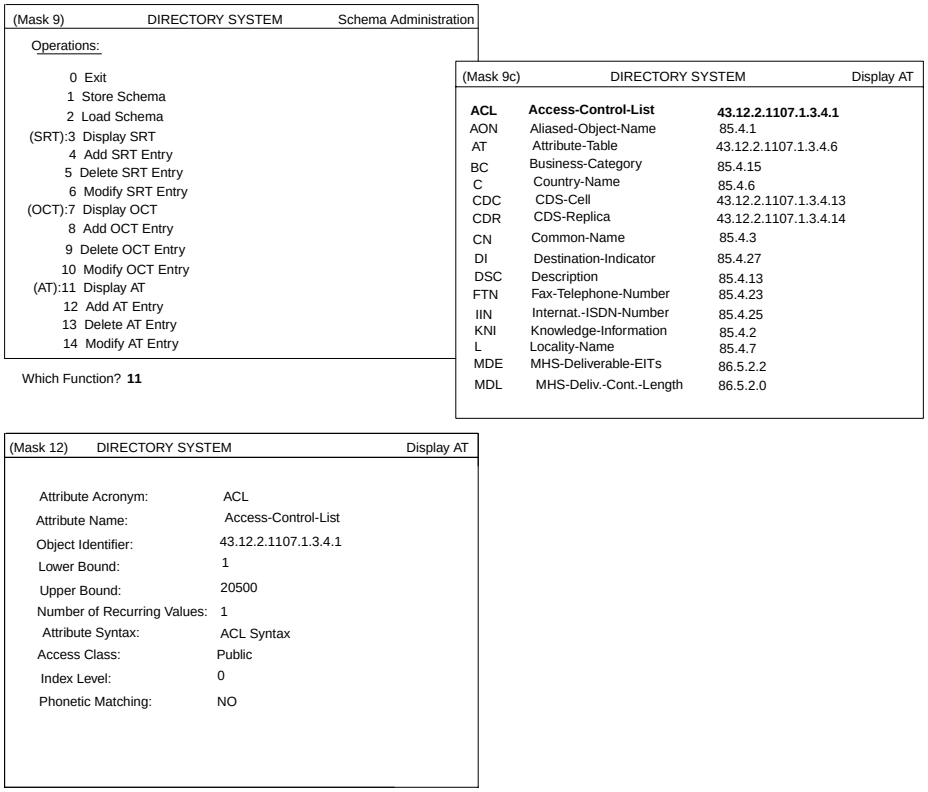
Mask 9 Select option number 11.

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 memory only. The **Store Schema** operation must then be called for changes in the DSA.

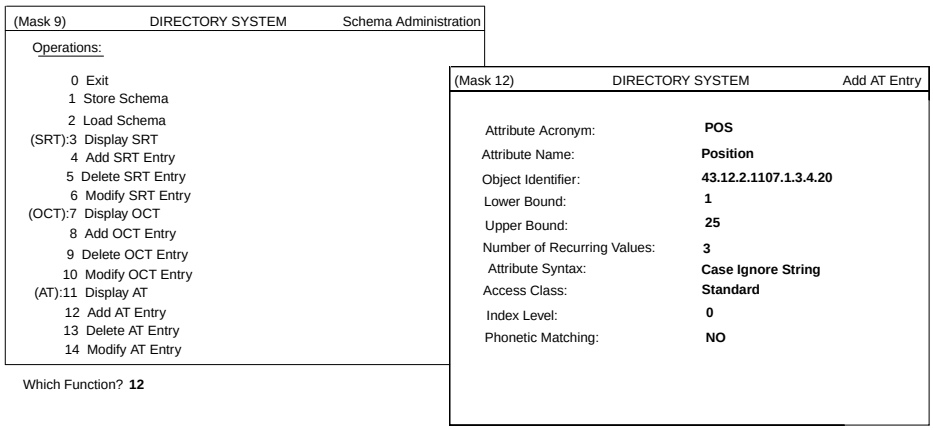
Mask Sequence

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 memory only. The **Store Schema** operation must then be called for changes in the DSA.

Mask Sequence

Mask 9 Select option number 13.

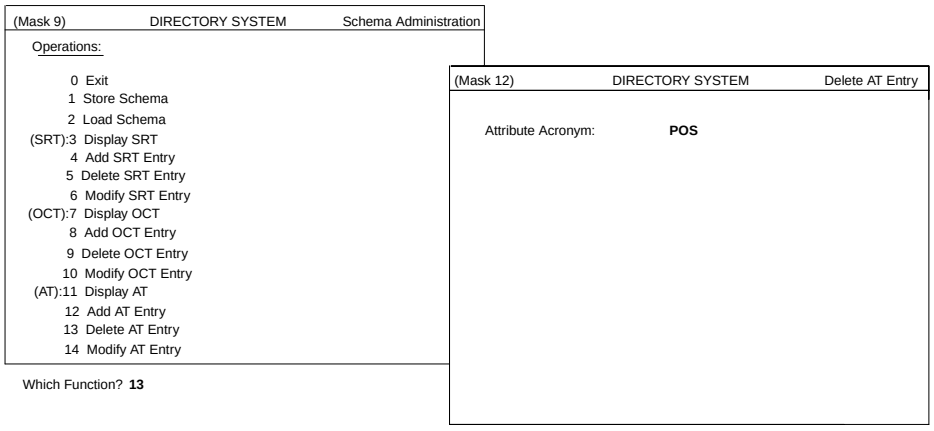
Mask 12 Enter the attribute acronym of the attribute to be deleted.

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.

Figure 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.

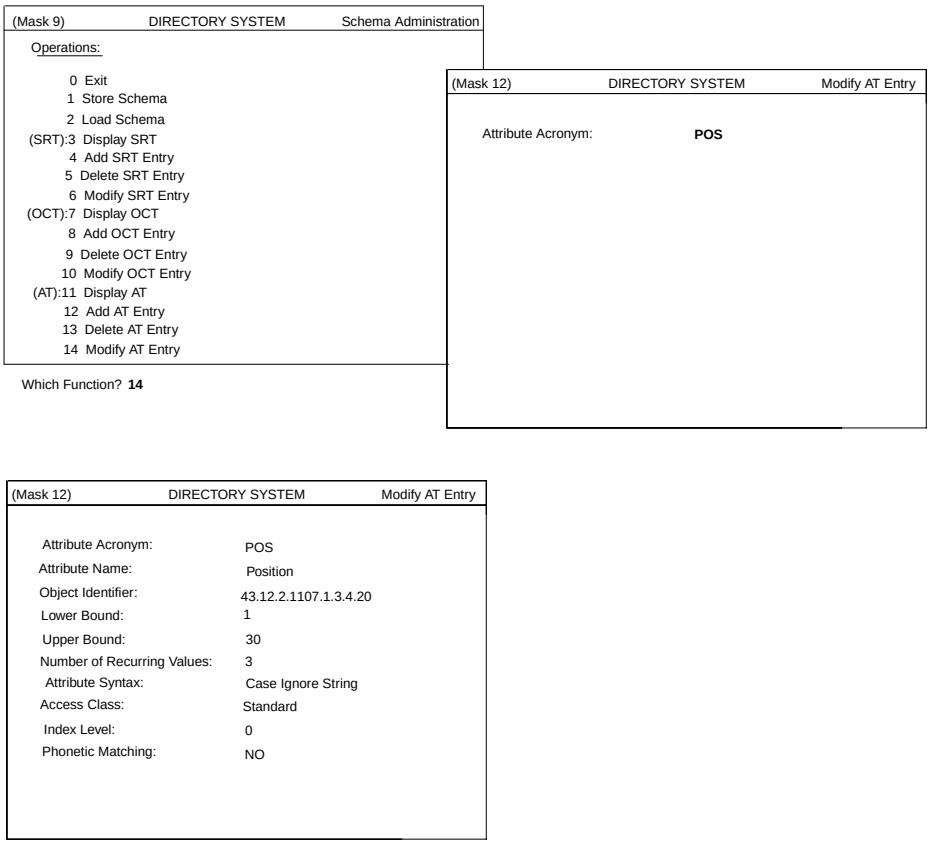
Mask 12 Only the **Attribute Acronym** field is displayed initially. When a correct attribute acronym is entered, Mask 12 is redisplayed with all values belonging to the AT entry. Fields can then be modified.

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. Administrators can manage shadows and shadowing jobs only for those master object entries for which they are 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 by using Masks 14g through 14j and can display errors that occurred during shadow updates by 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

```
(Mask 2)                              DIRECTORY SYSTEM                              operation


DSA IDENTIFICATION:
Country-Name:       - -
Organization-Name:  - - - - - - - - - - - - -
Org.-Unit-Name:     - - - - - - - - - - - - -
Common-Name:        - - - - - - - - - - - - -
Common-Name:        - - - - - - - - - - - - -
    Options: None
```

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 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. (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

```
(Mask 5)                              DIRECTORY SYSTEM                                operation



Structure Rule
Name Structure  01 .............................    ...............................
02 .............................    ...............................
03 .............................    ...............................
04 .............................    ...............................
05 .............................    ...............................
06 .............................    ...............................
07 .............................    ...............................
08 .............................    ...............................
09 .............................    ...............................
10 .............................    ...............................
11 .............................    ...............................
12 .............................    ...............................
```

```
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	

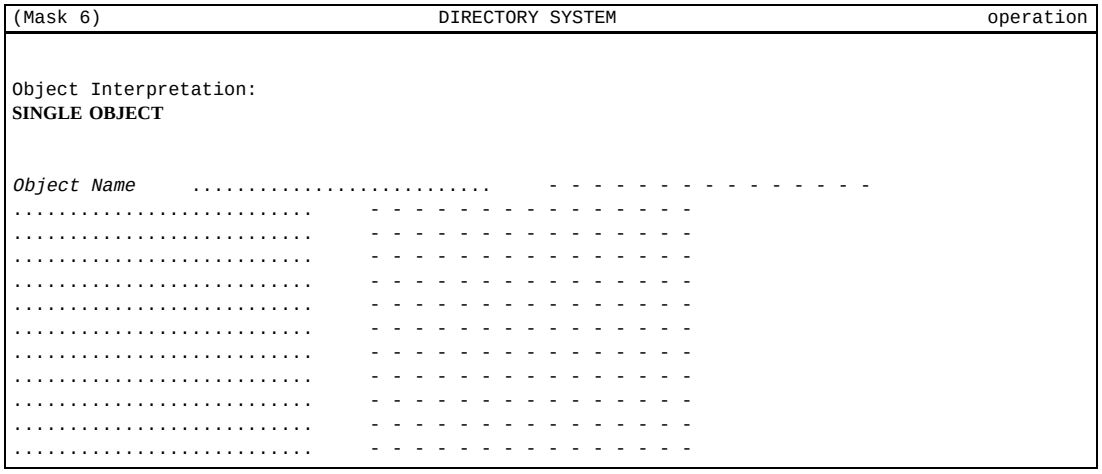
```
Which Structure Rule ?
```

If the administration function **Shadow Administration** is selected in Mask 3 (see Figure 7-4), the system also displays the structure rule **00 ROOT** in Mask 5. The **00 ROOT** 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 the **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)

```
(Mask 14)                          DIRECTORY SYSTEM                          operation

  Object Name:           ........................................
........................................
  Object Interpretation:
interpretation
  Target DSA:            ........................................
........................................
  Job:
job
  Update Frequency:
HIGH
```

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

```
(Mask 14)                          DIRECTORY SYSTEM                          operation

  Object Name:           ........................................
.........................................
  Object Interpretation:
interpretation
  Target DSA:            ........................................
.........................................
  Job:
job
  Update Frequency:      HIGH
  Update Times :
EVERY 5 MINUTES
```

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

```
(Mask 14)                          DIRECTORY SYSTEM                          operation

  Object Name:           ........................................
........................................
  Object Interpretation:
interpretation
  Target DSA:            ........................................
........................................
  Job:
job
  Update Frequency:      LOW
  Update Times :
EVERY DAY
```

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

```
(Mask 14)                          DIRECTORY SYSTEM                          operation

  Object Name:            ....................................
.....................................
  Object Interpretation:
interpretation
  Target DSA:             ....................................
.....................................
  Job:                    NOT ACTIVE
```

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:

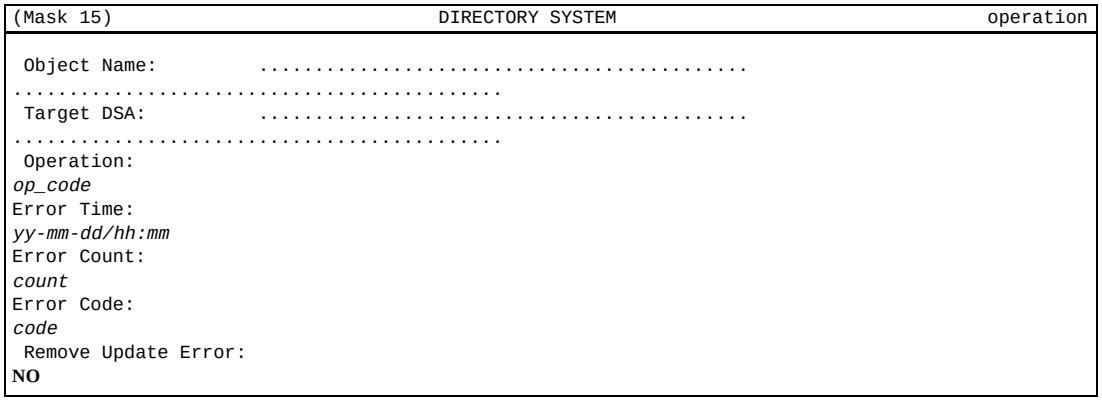
SINGLE OBJECT
OBJECT AND ITS SUBORDINATES

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



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.

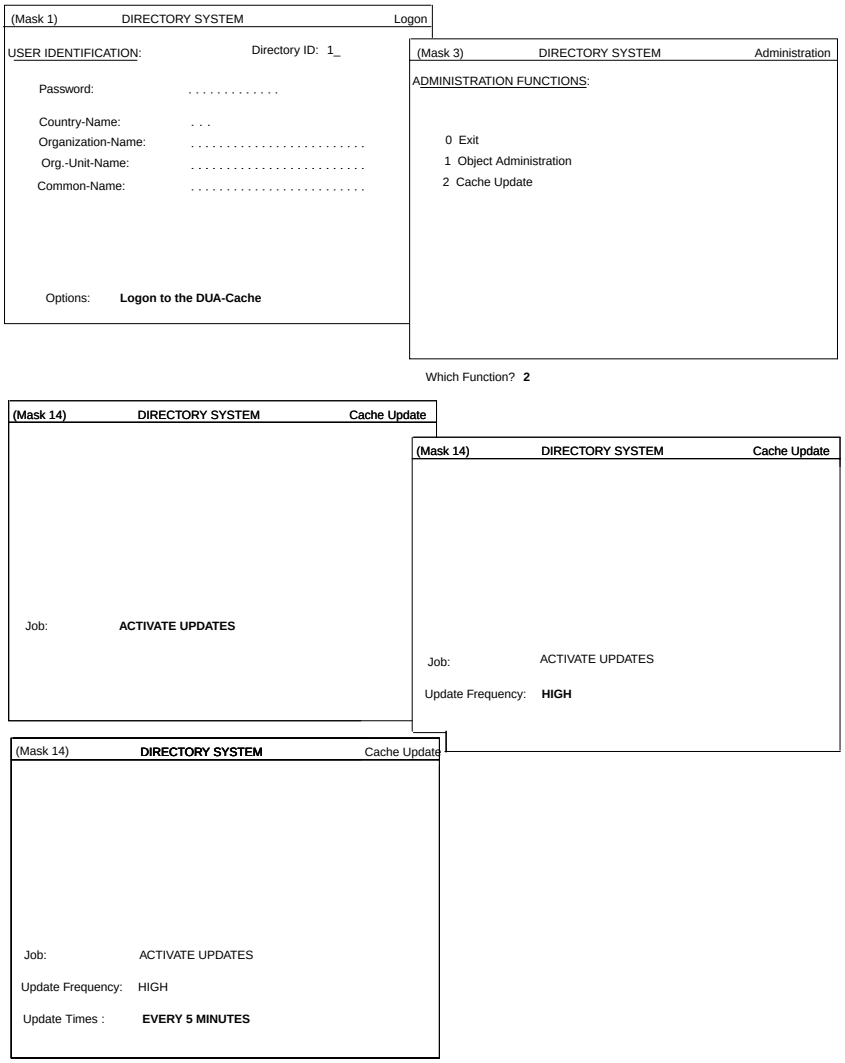
Mask 14e Enter the frequency in days per week.

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 5 minutes. Administrator's selections are highlighted in bold.

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).

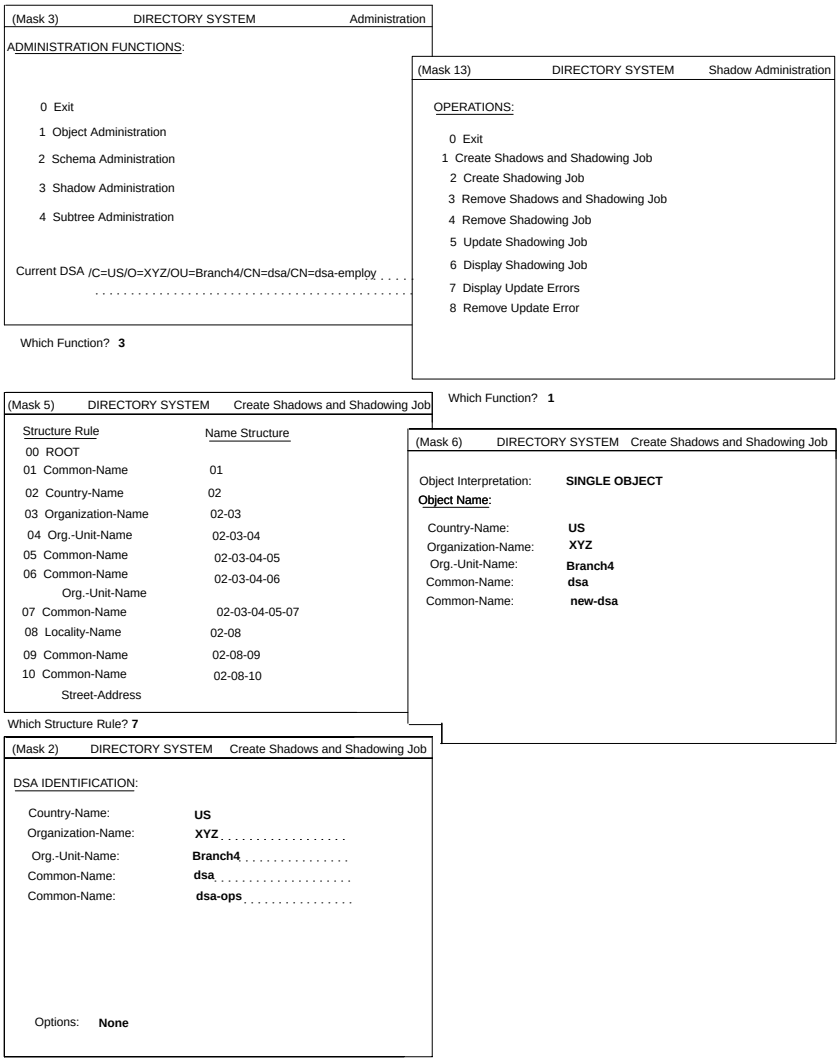
You are then returned to Mask 2, in which a further target DSA name can be specified. If no more DSAs have to be specified, press **<Del>**.

Figure 10-18 shows a sample mask sequence where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object for which the shadow is created.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA on which the shadow is created.
- The shadowing job is already set to an active state with **MEDIUM** update frequency (**EVERY 2 HOURS**).

Administrator's selections are highlighted in bold.

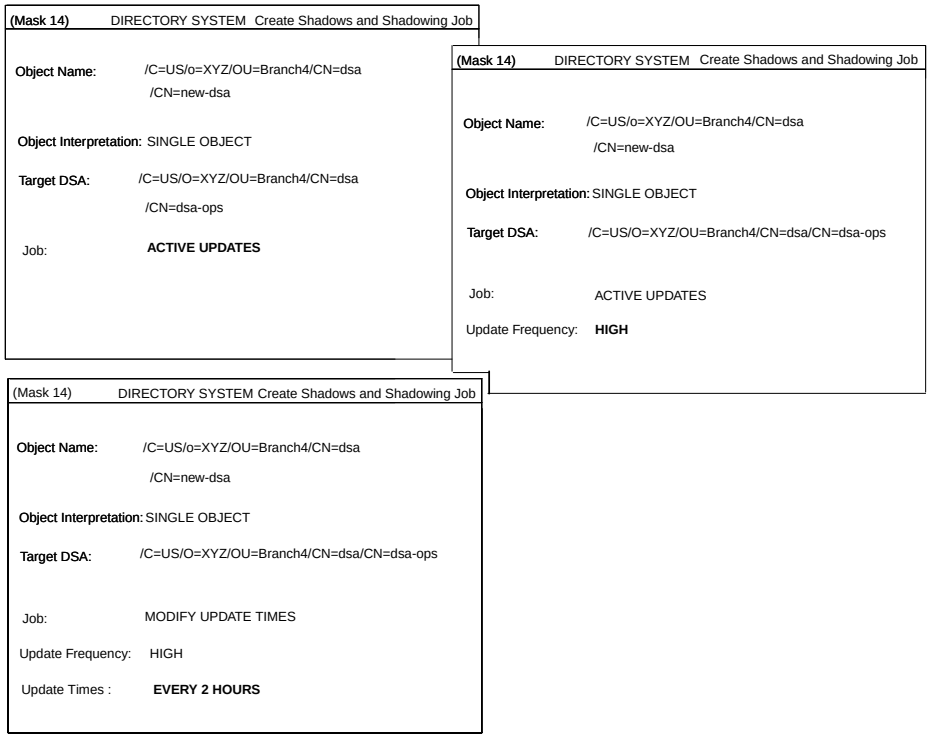
Figure 10–18. Sample Create Shadows and Shadowing Job Operation (Part 1)



After Mask 2 in Figure 10-18, Mask 14 is displayed (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 administrator 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).

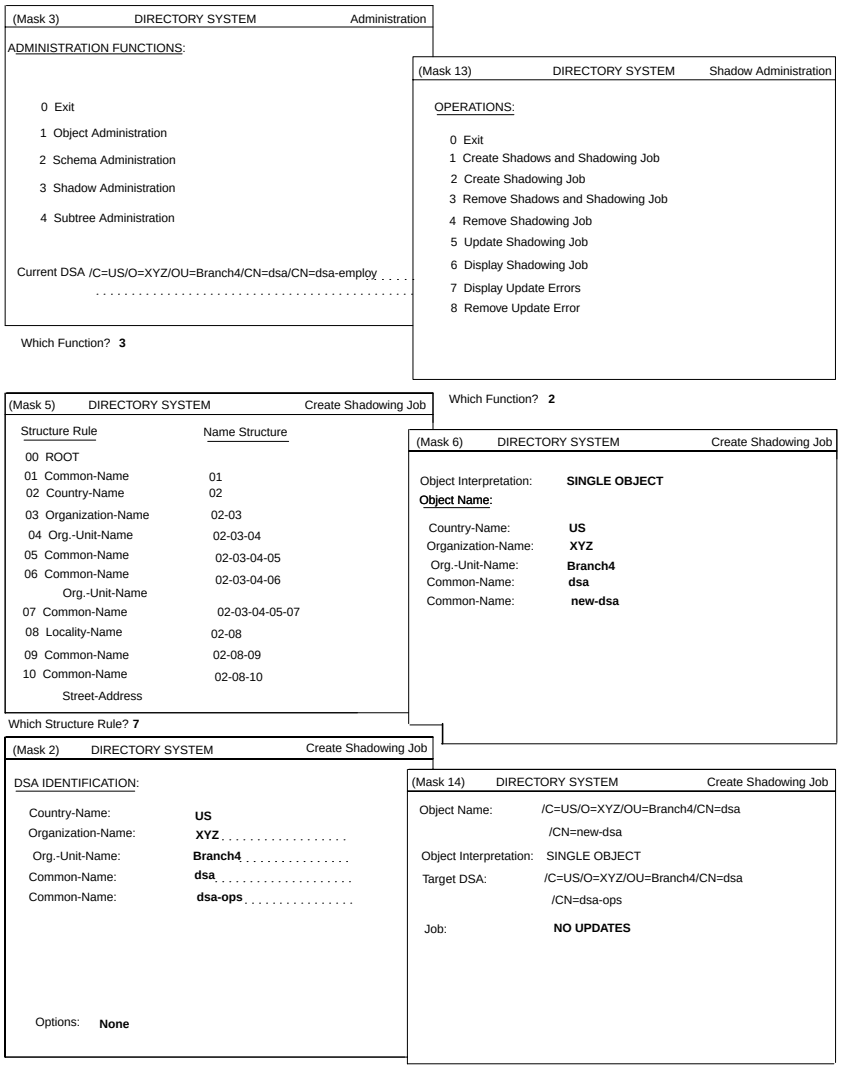
You are then returned to Mask 2, in which a further target DSA name can be specified. If no more DSAs have to be specified, press **<Del>**.

Figure 10-20 shows a sample mask sequence where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object for which an inactive shadowing job is created.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA.

Administrator's selections are highlighted in bold in the figure.

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.

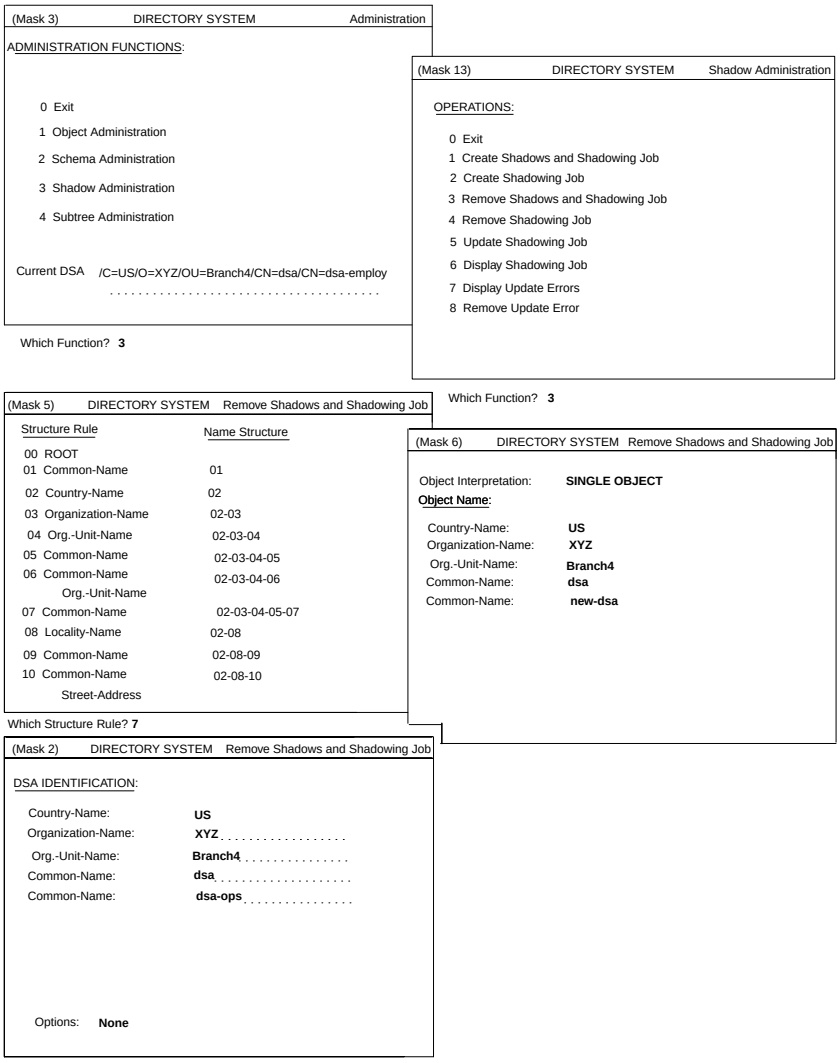
Figure 10-21 shows a sample mask sequence for Mask Sequence 2, which directly specifies a shadowing job to be removed, where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN for the object whose shadowing job is to be removed.

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA from which the shadow is removed.

Administrator's selections are highlighted in bold.

Figure 10–21. Sample Remove 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.

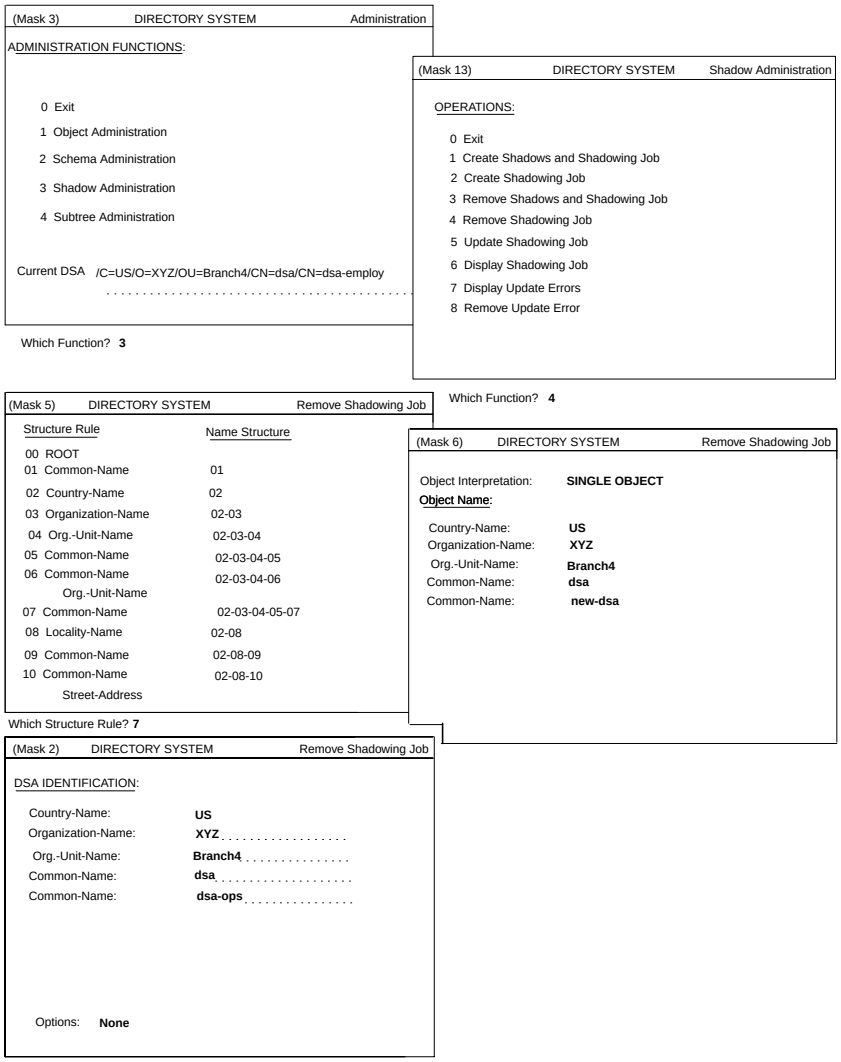
Figure 10-22 shows a sample mask sequence for Mask Sequence 2, which directly specifies a shadowing job to be removed, where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object whose shadowing job is to be removed.

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA from which the shadow is removed.

Administrator's selections are highlighted in bold.

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, which updates a shadowing job, where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object whose shadowing job is to be updated.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA on which the shadowing job is to be updated.

The new update frequency is every 5 minutes. Administrator’s selections are highlighted in bold.

Figure 10–23. Sample Update Shadowing Job Operation (Part 1)

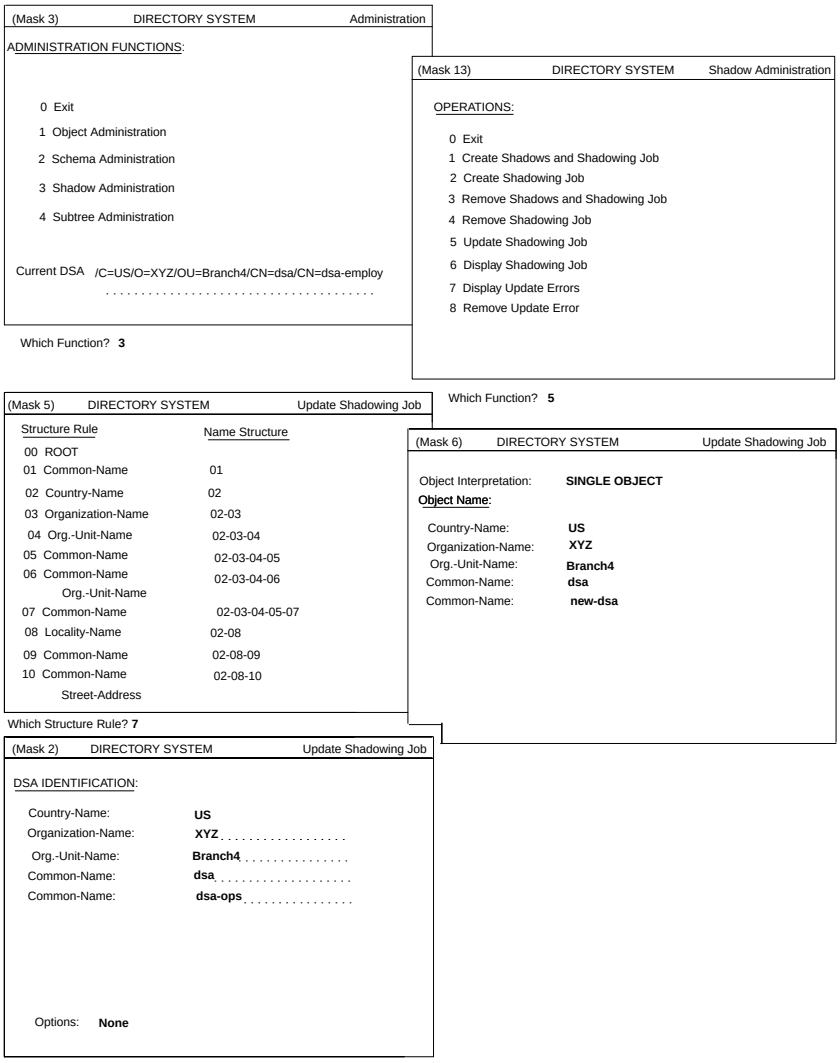
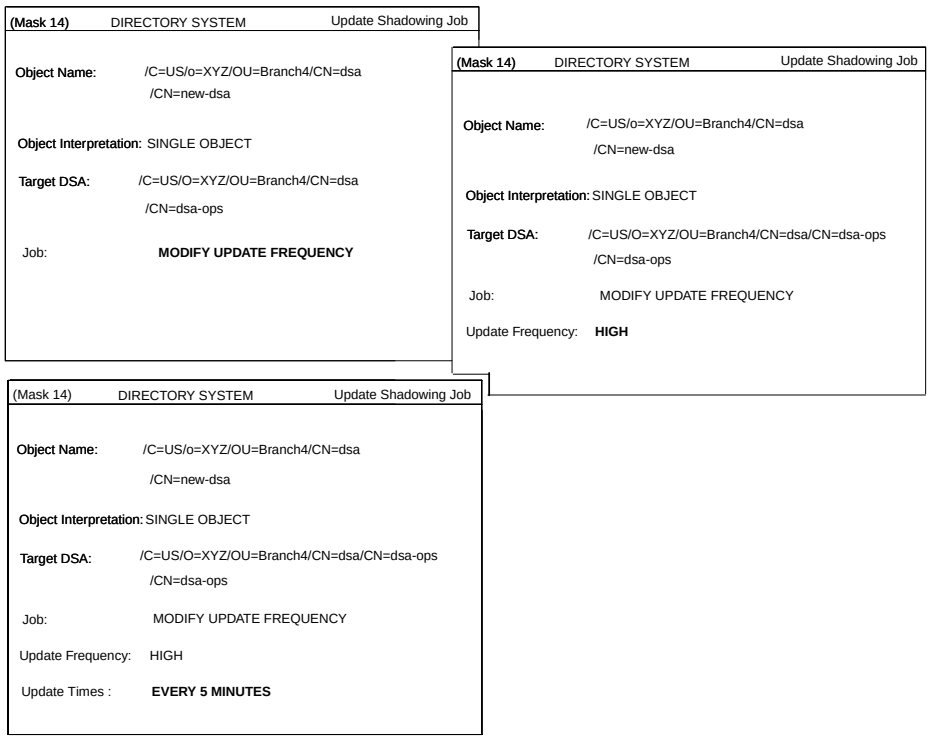


Figure 10–24. Sample Update Shadowing Job Operation (Part 2)



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**.

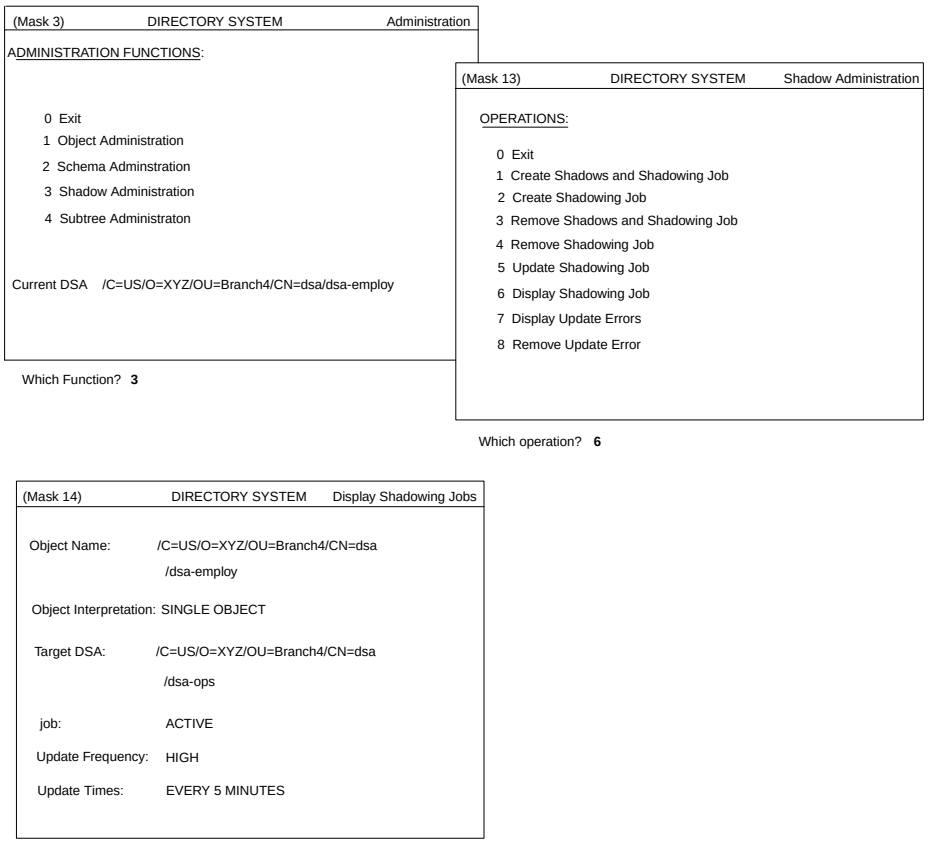
If no job is to be marked, terminate the operation by pressing **<Del>**.

Figure 10-25 shows a sample mask sequence that displays an active shadowing job with update frequency set to **HIGH** (**EVERY 5 MINUTES**), where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object for which the shadowing job is set.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA on which the shadowing job is set.

Administrator's selections are highlighted in bold.

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>**.

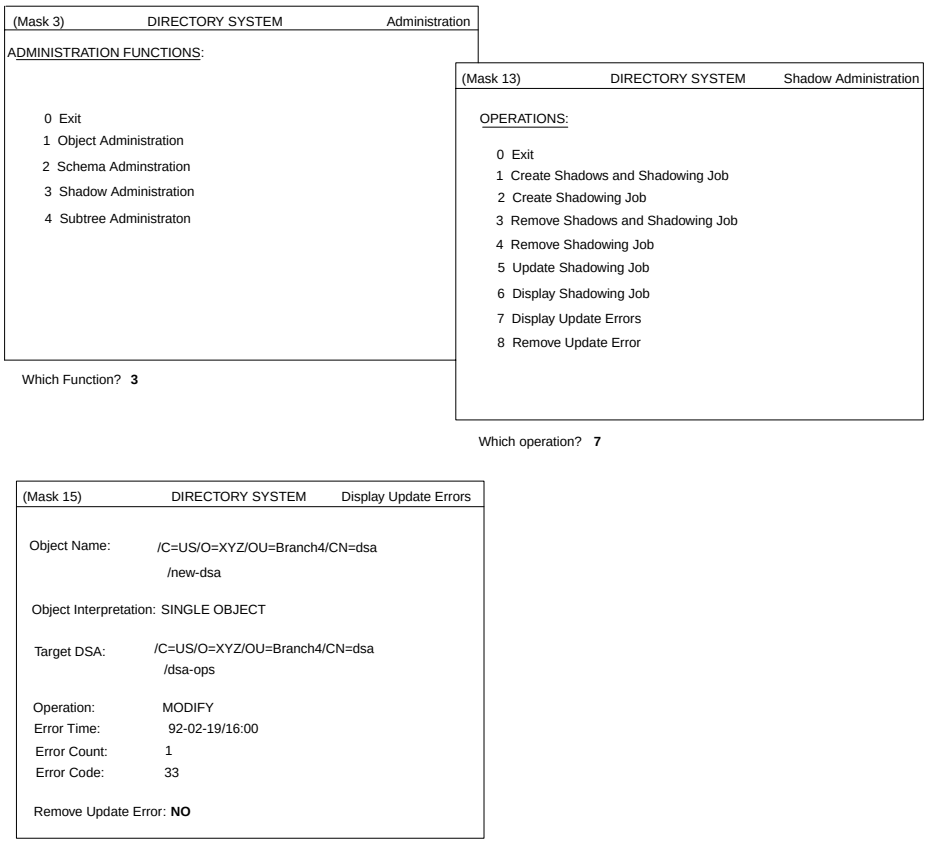
Terminate the operation by pressing **<Del>**.

Figure 10-26 shows a sample mask sequence where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object whose shadowing job update errors are to be displayed.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA.

Administrator's selections are highlighted in bold.

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.

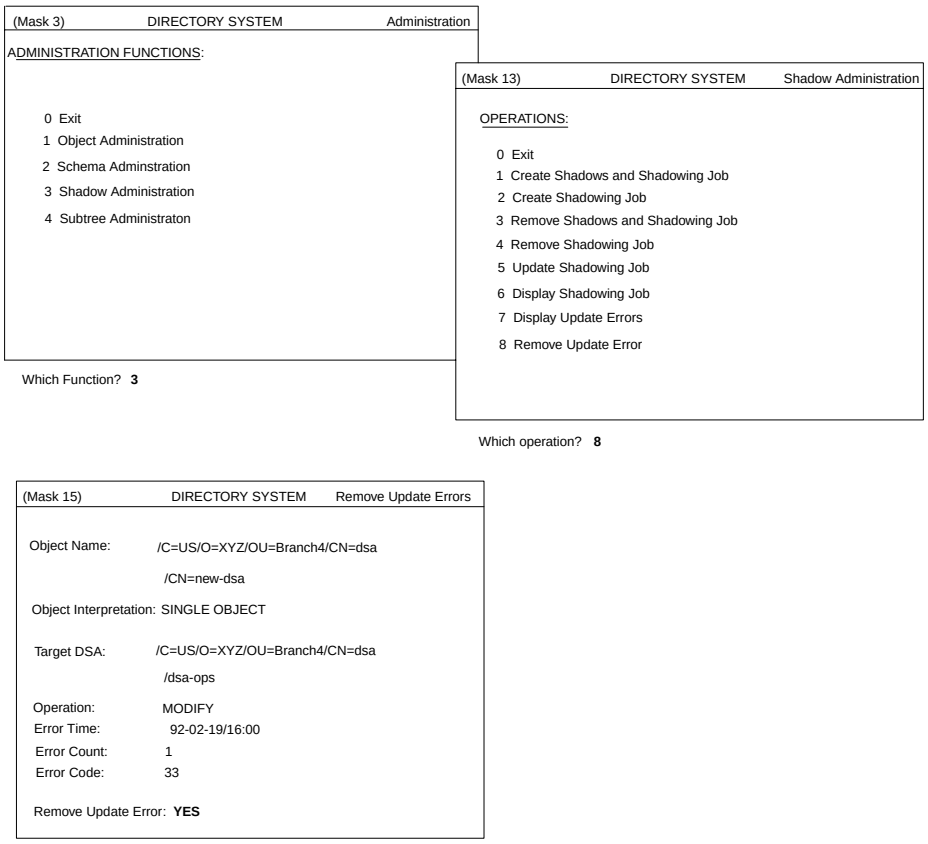
If **Remove Update Error** is set to **YES**, this update error is ignored by the delta update process at the next activation time (otherwise, the delta update process tries to resend the erroneous update operation to the target DSA).

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. Administrator's selections are highlighted in bold.

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 by 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 by 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 by 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 11-2 illustrates how a sequence of **Save Subtree** and **Append Subtree** works when a subtree is saved and appended under the same parent node.

Figure 11–1. Moving a Subtree

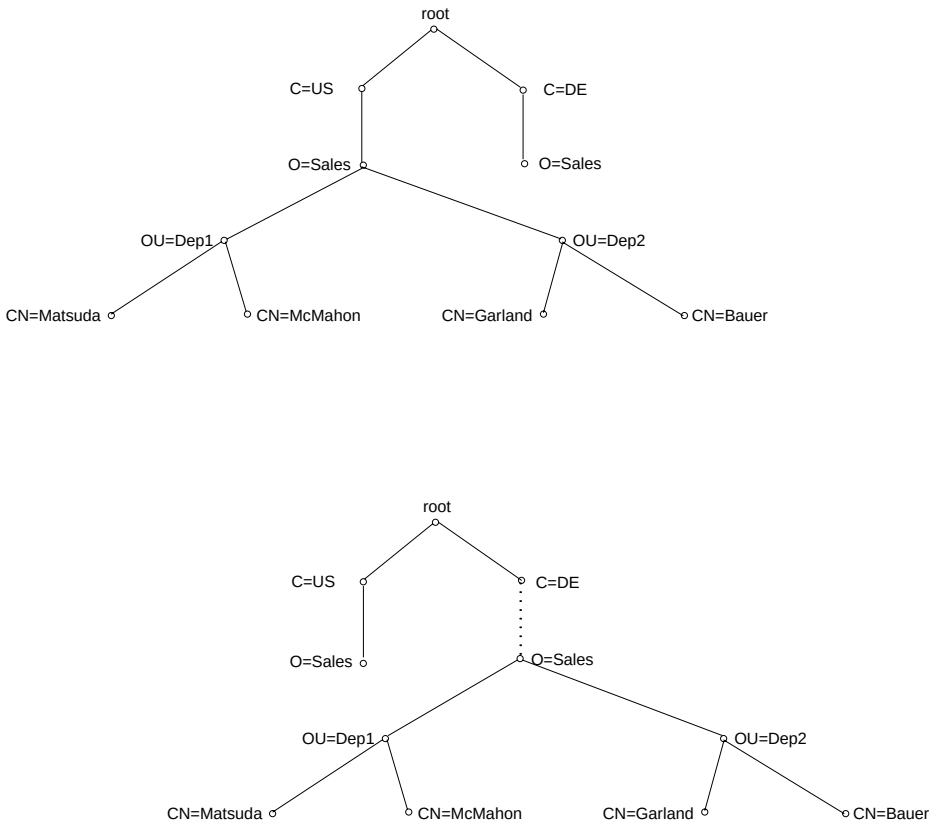
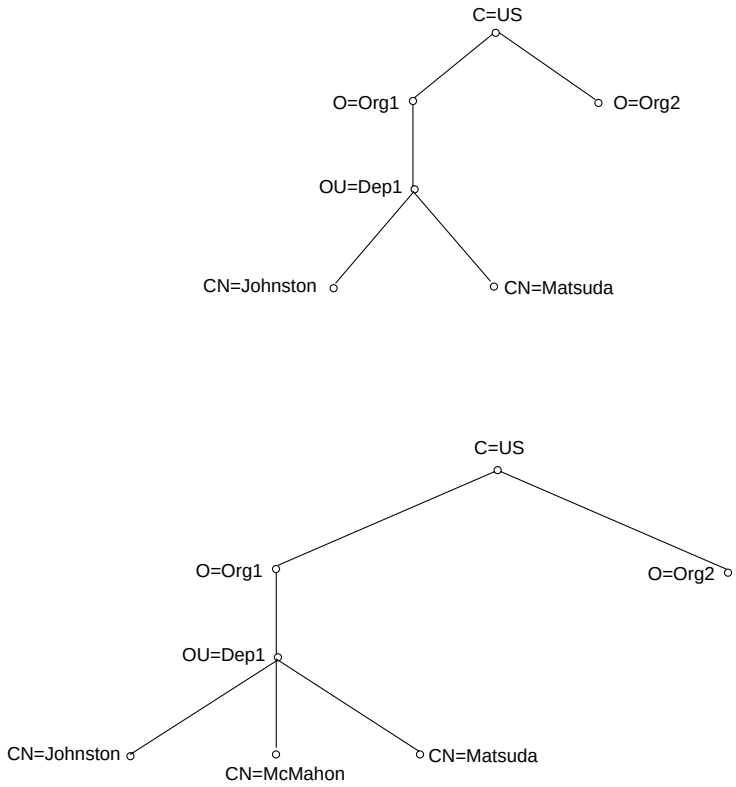


Figure 11–2. Appending a Subtree Under the Same Parent Node



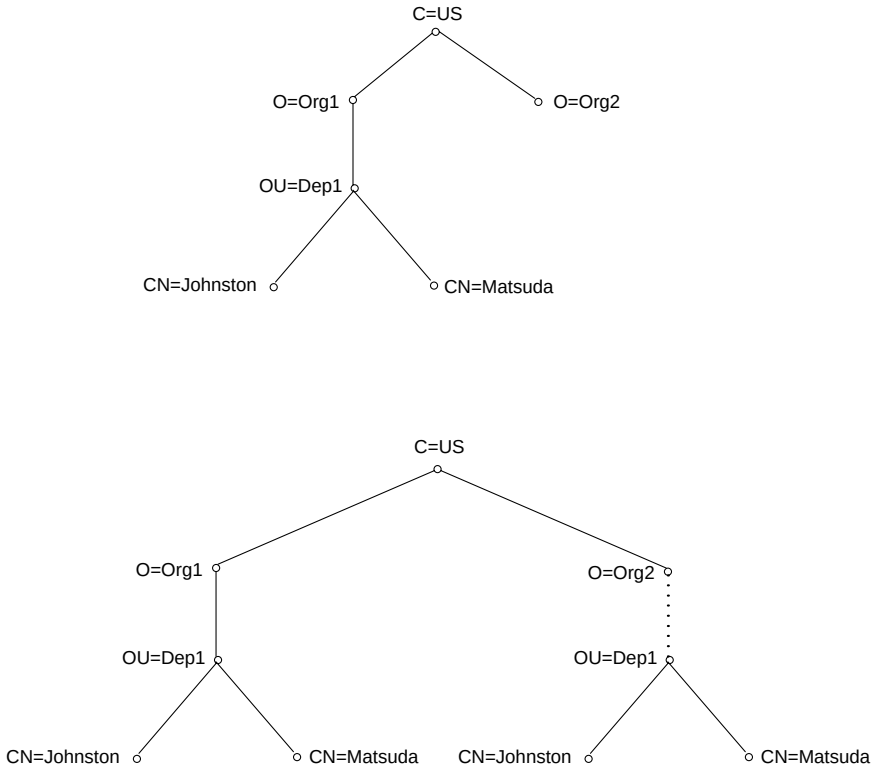
In Figure 11-2, the following occur:

- 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 the following DNs take their **Master-Knowledge** attributes from the save file:
  - **/C=US/O=Org1/OU=Dep1/CN=Johnston**

— **/C=US/O=Org1/OU=Dep1/CN=Matsuda**

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

```
(Mask 2)                              DIRECTORY SYSTEM                              operation


DSA IDENTIFICATION:    Country-Name:       - -
Organization-Name:  - - - - - - - - - - - - -
Org.-Unit-Name:     - - - - - - - - - - - - -
Common-Name:        - - - - - - - - - - - - -
Common-Name:        - - - - - - - - - - - - -
Options: None
```

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

```
(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 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 ?
```

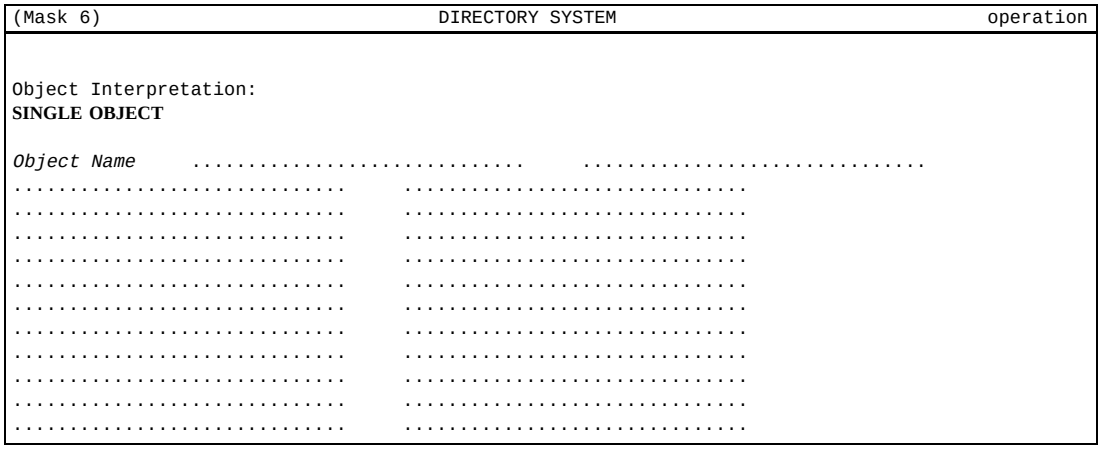
The **00 Root** rule is not displayed if the structure rule for the DN of **Target DSA** or **Source DSA** must be entered.

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)

```
(Mask 8)                              DIRECTORY SYSTEM                                operation


Attribute:    Name:      - - - - - - - - - - - - -
    Old Value: - - - - - - - - - - - - -
- - - - - - - - - - - - -
    New Value: - - - - - - - - - - - - -
- - - - - - - - - - - - -
```

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 the operation is **Modify Subtree**.

Old Value Enter the old value of the attribute. This value is only valid if the operation is **Modify Subtree**.

New Value If the 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 the 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)

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)

```
(Mask 17a)                         DIRECTORY SYSTEM                          operation


Source DSA:
MASTER DSA(s)
  File Name:    - - - - - - - - - - - - -
- - - - - - - - - - - - -
```

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)

```
(Mask 17b)                          DIRECTORY SYSTEM                                operation

  File Name:                 - - - - - - - - - - -
- - - - - - - - - - -
  Overwrite Existing Entries: YES
  New Entries Protected By:
ACL of the new parent
  Target DSA:
BIND DSA
```

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 the 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**

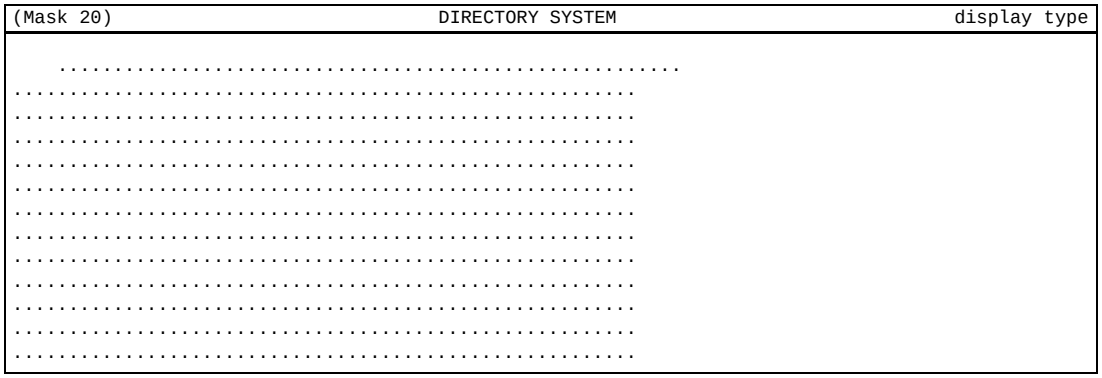
In the case of the **Copy Subtree** operation, the **File Name** field is not displayed in the mask.

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.

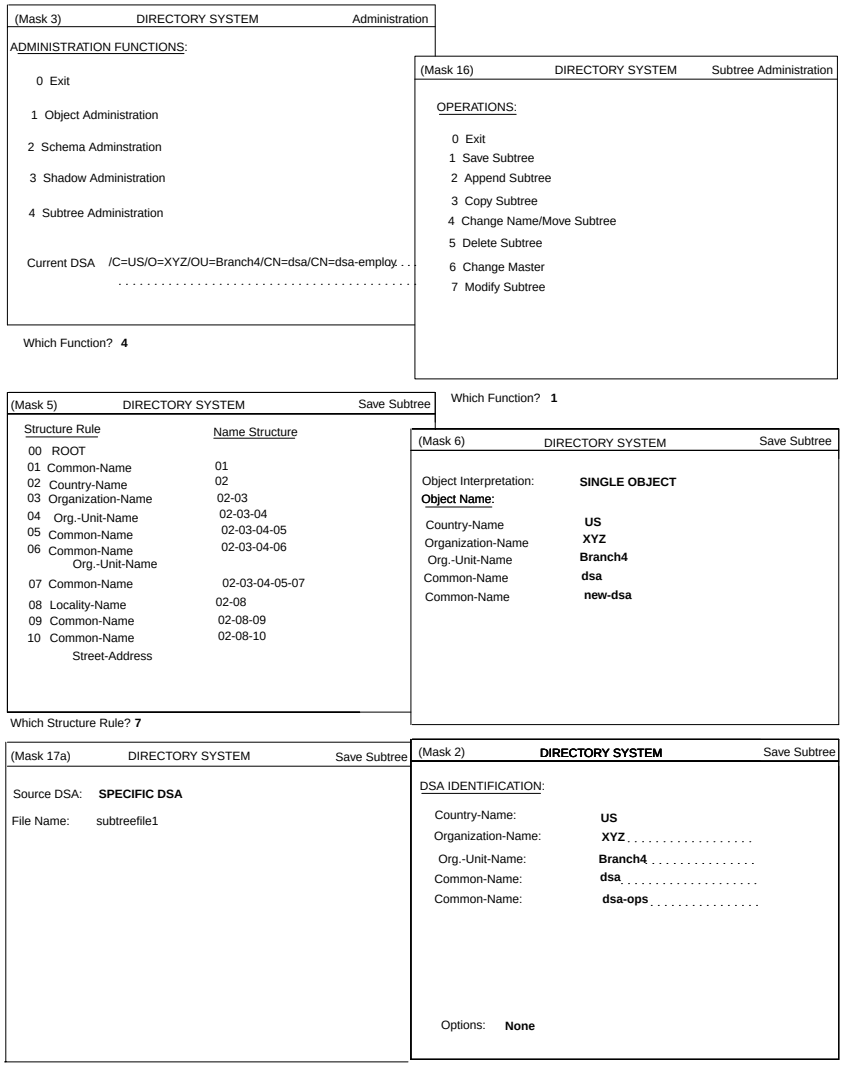
Mask 2 Enter the DN of the DSA.

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**, where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the source DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.

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 (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.

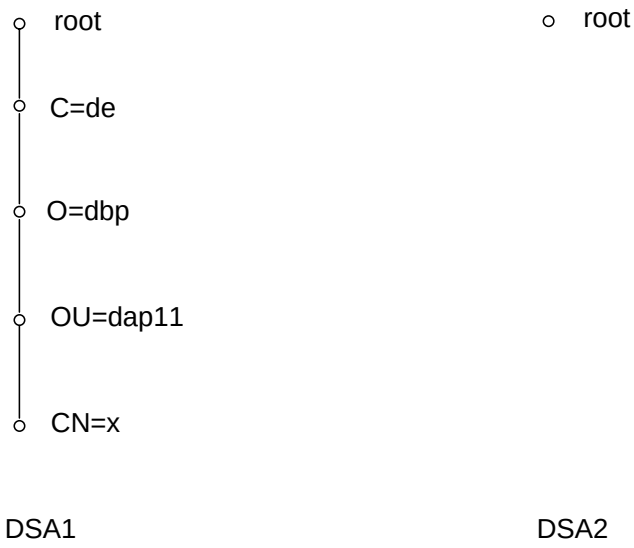
If the subtree is to be appended under the same parent node from where it is saved, objects to be appended that already exist keep the same **Master-Knowledge** attribute value. Objects to be appended that do not exist take the value of the **Master-Knowledge** attribute from the save file.

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.

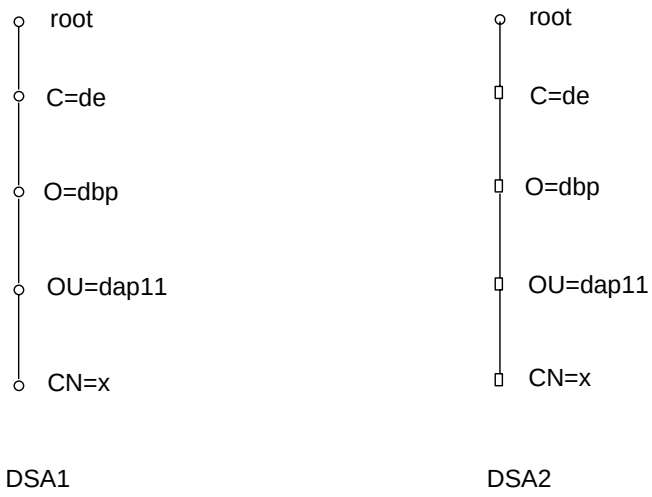
Figure 11–14. DSA1 and DSA2 Before Save and Append Operations



- ○ master entries
- ▯ shadow entries

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



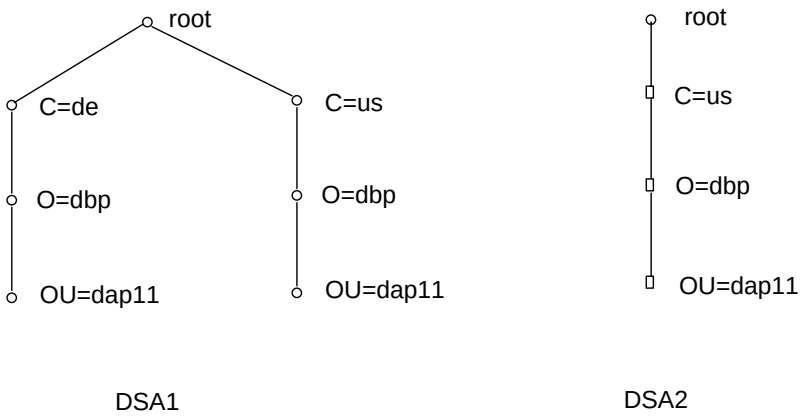
- ○ master entries
- ▯ shadow entries

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



- ○ master entries
- ▯ shadow entries

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.

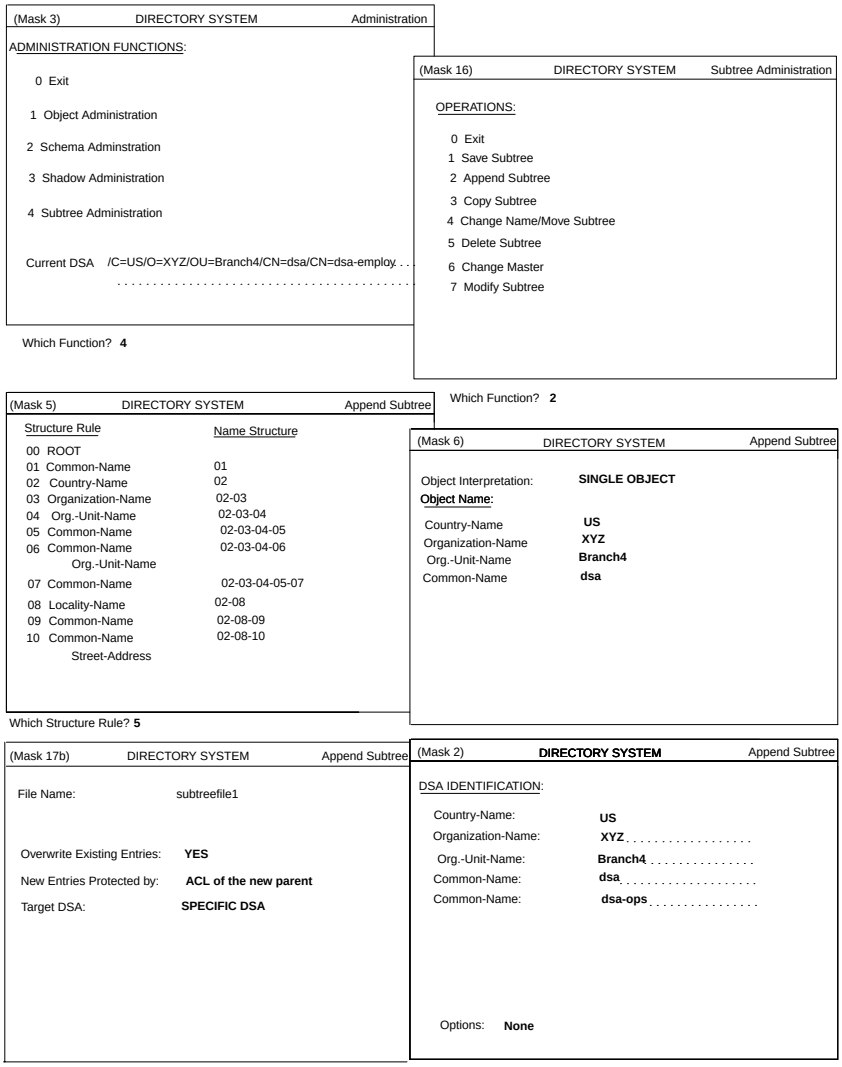
If an object to be added during this operation already exists, but has different attributes, the object is displayed in Mask 20. Depending on the selection in the **Overwrite Existing Entries** field, the *display type* field in Mask 20 is either **Nonoverwritten objects** or **Overwritten objects**.

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**, where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the target DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.

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 11.2.1 and 11.2.2). 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 to the current DSA where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the source DSA.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.

The administrator's selections are highlighted in bold.

Figure 11–18. Sample Copy Subtree Operation (Part 1)

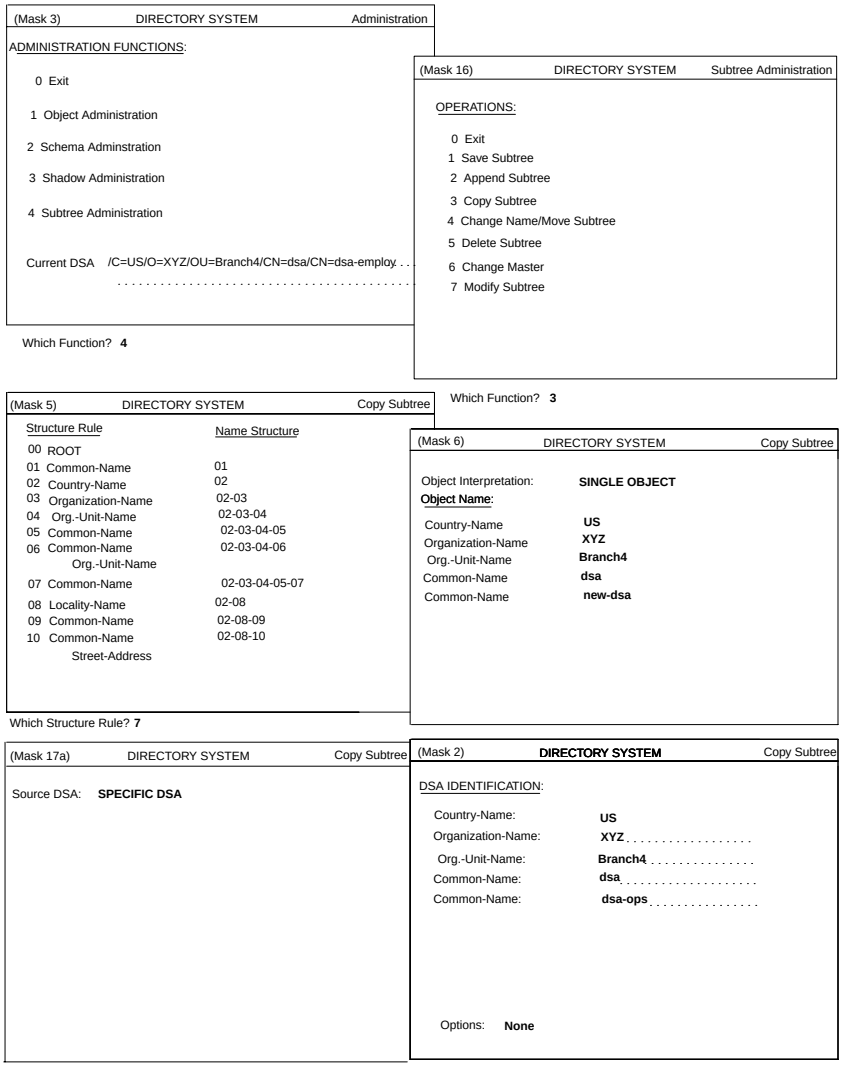
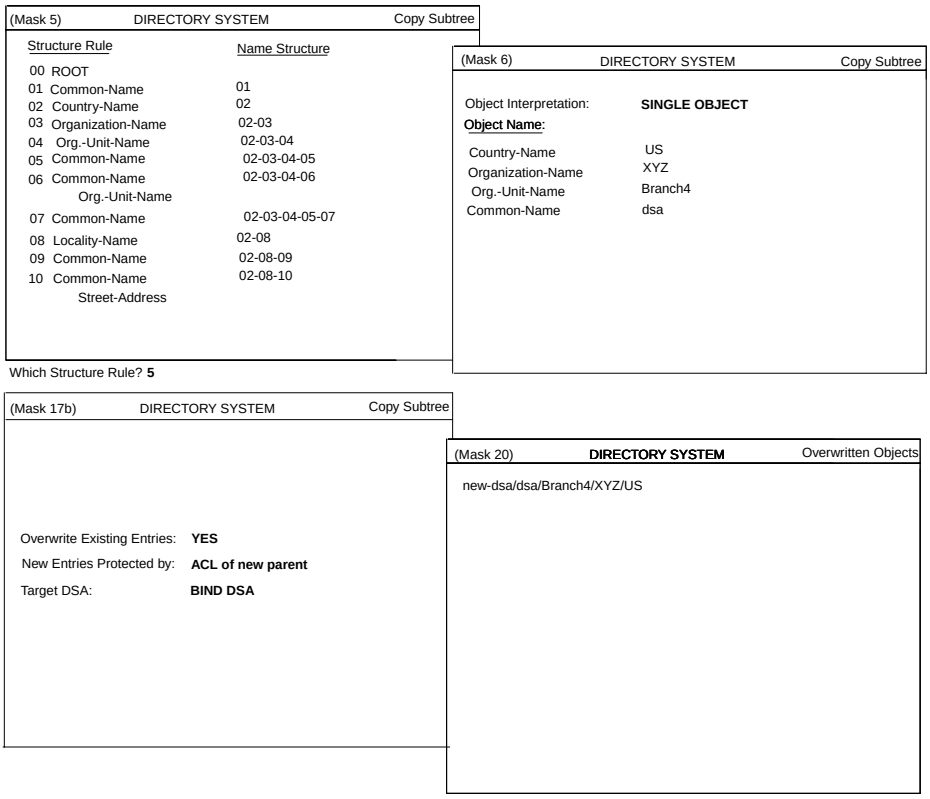


Figure 11–19. Sample Copy Subtree Operation (Part 2)



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 at **/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

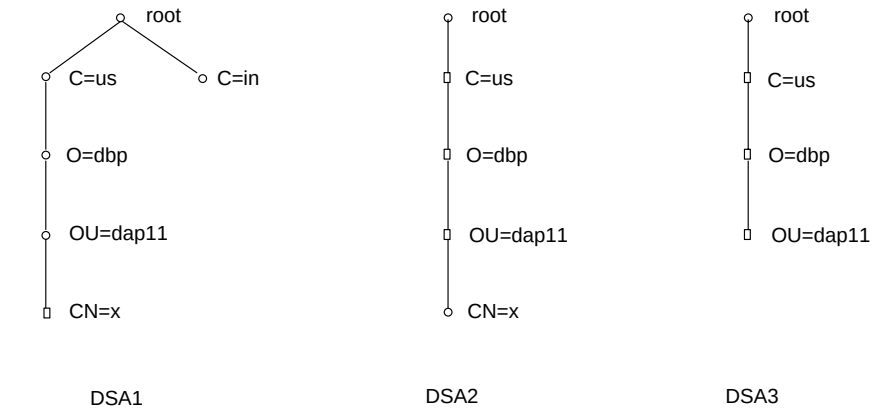
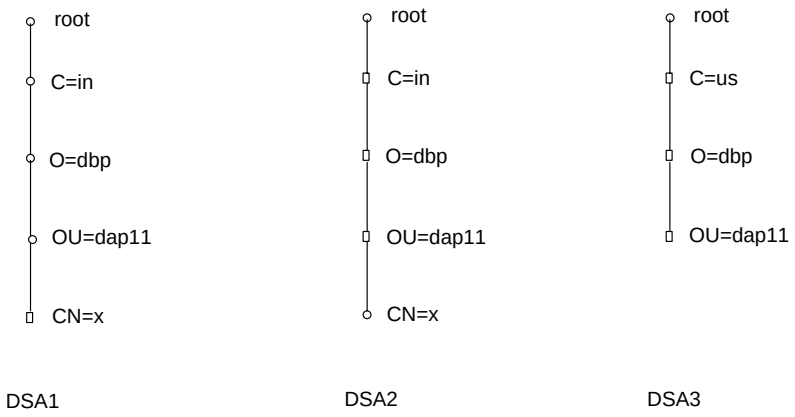


Figure 11–21. After a Change Name/Move Subtree Operation



○ master entries

▯ shadow entries

The shadowing job for **/C=us** (and its subordinates) for **DSA2** will fail. The administrator should run the shadowing job for **DSA3** and 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 an object entry as follows:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the object's original DN.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-doc** is the object's new (changed) DN.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.

The administrator's selections are highlighted in bold.

Figure 11–22. Sample Change Name/Move Subtree Operation (Part 1)

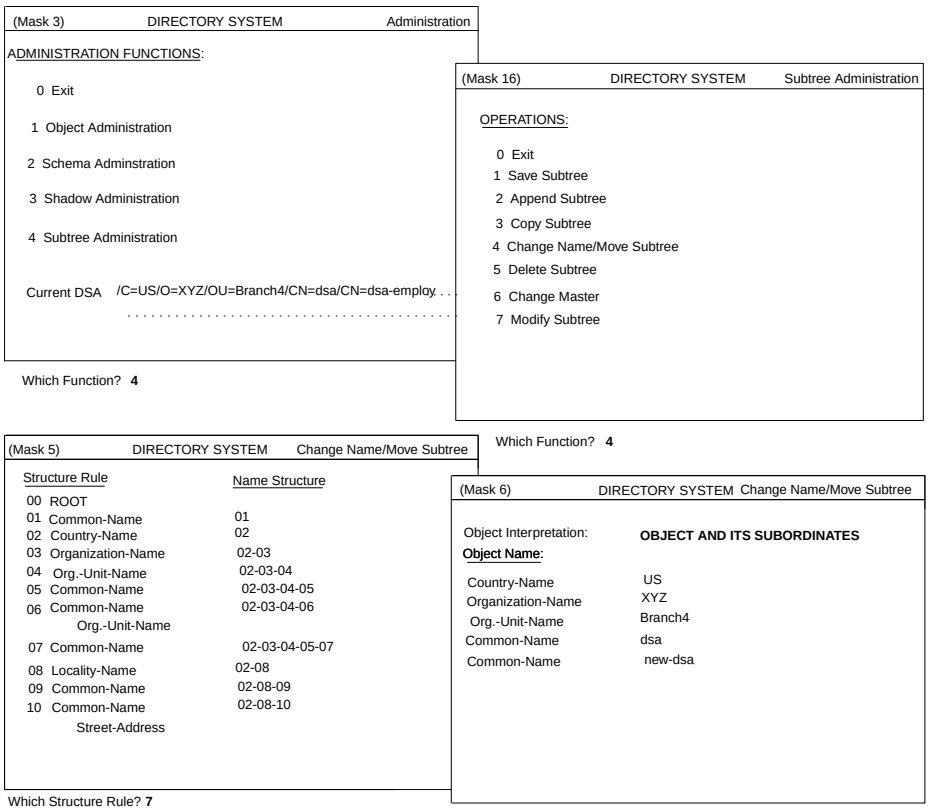
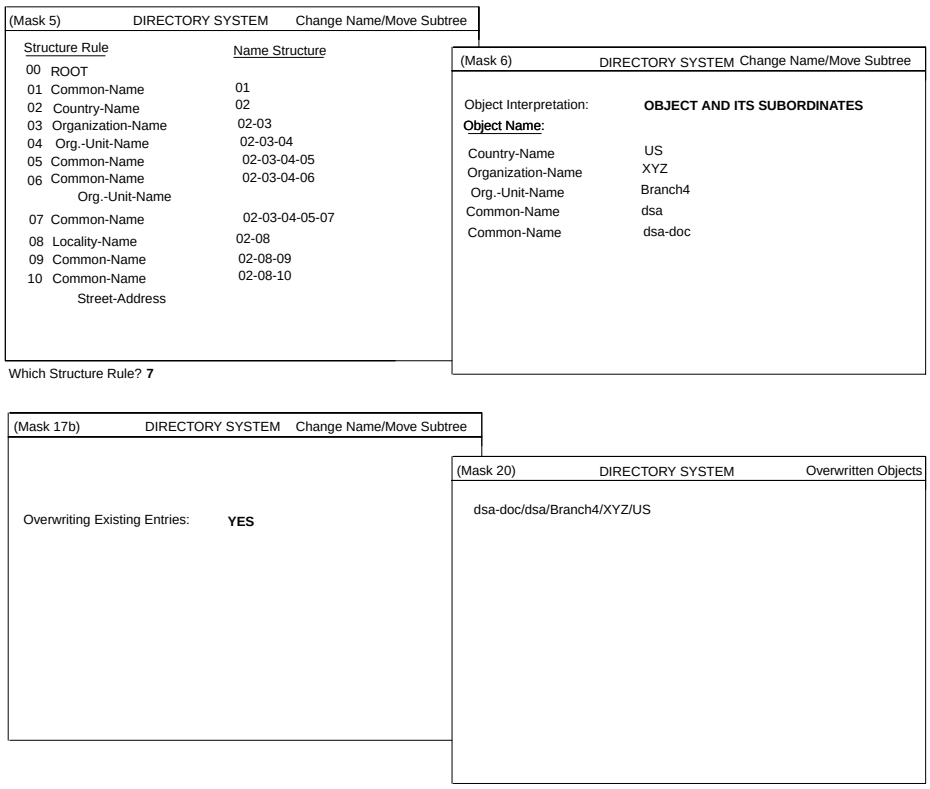


Figure 11–23. Sample Change Name/Move Subtree Operation (Part 2)



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 at **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-doc** (the new name).

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.

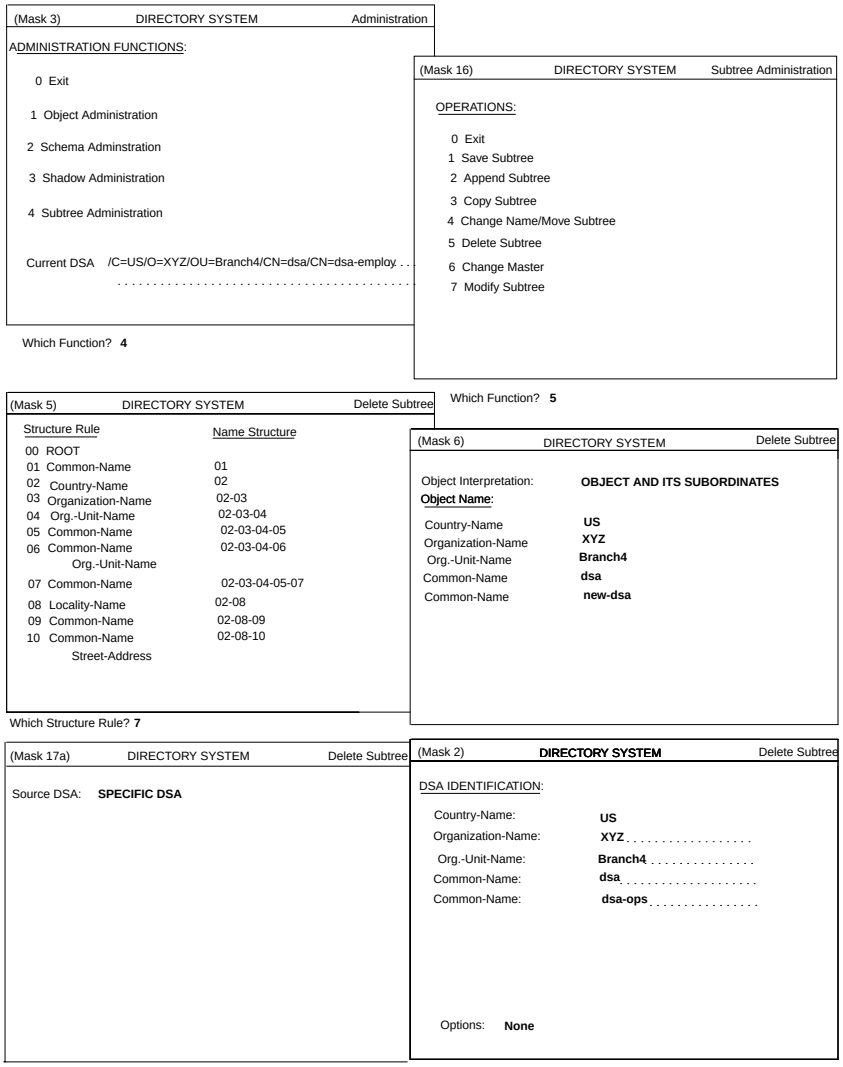
The **File Name** field is not displayed in Mask 17a.

Figure 11-24 shows a sample mask sequence that deletes the name of an object entry where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the source DSA, from which the object is being deleted.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the current DSA.

The administrator's selections are highlighted in bold.

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. Before a Change Master Operation

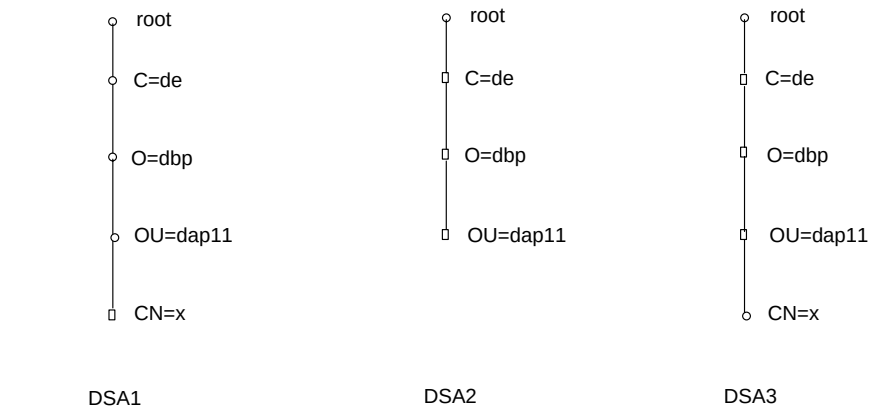
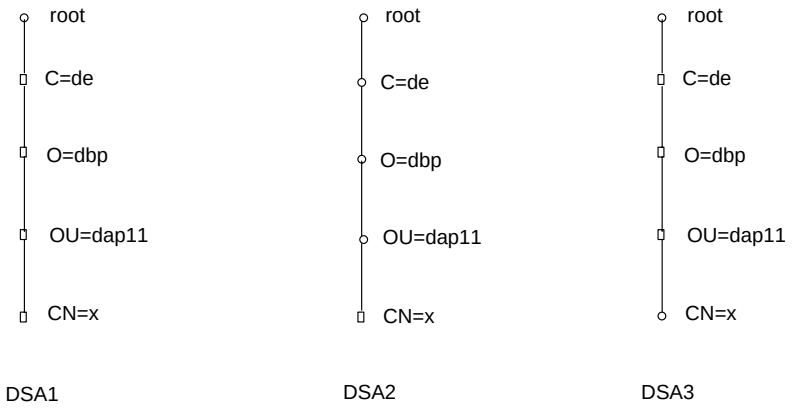


Figure 11–26. After a Change Master Operation



- ○ master entries
- ▯ shadow entries

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.

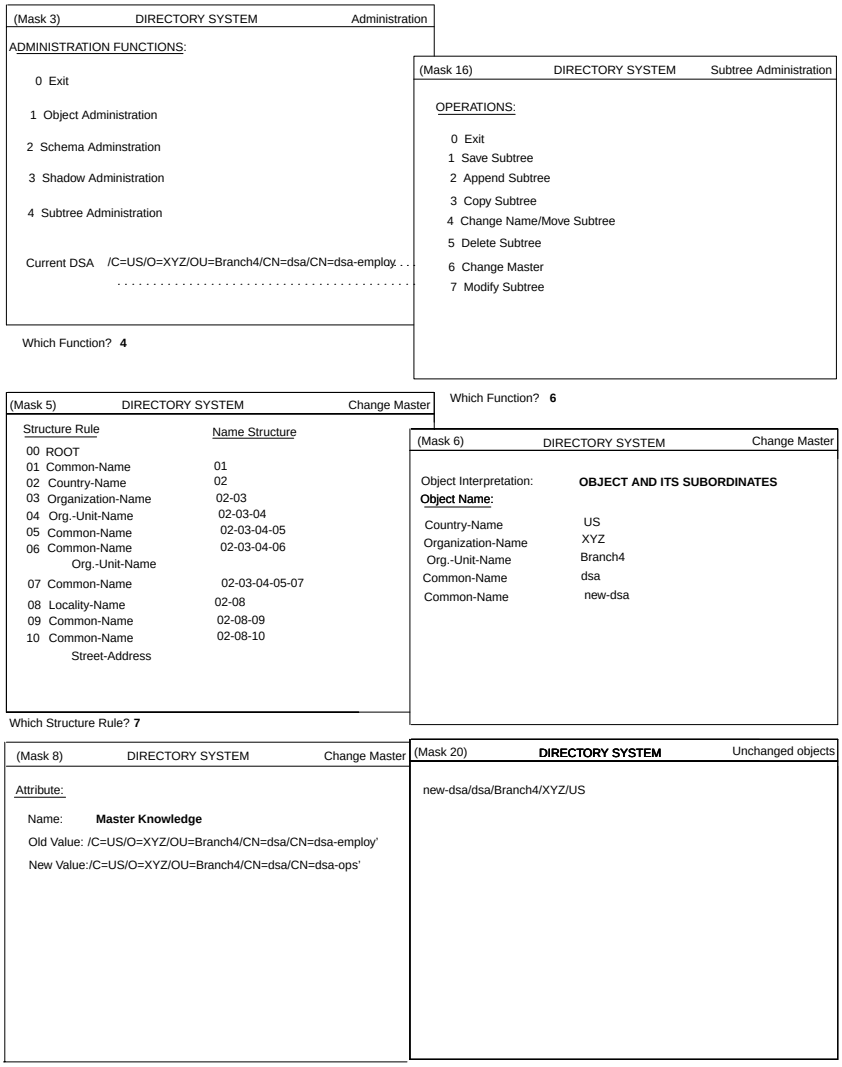
If any problems arise during this operation (for example, due to incorrect access rights), a list of names that cannot be changed is displayed in Mask 20.

Figure 11-27 shows a sample mask sequence that changes the **Master-Knowledge** attribute of an object entry where:

- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=new-dsa** is the DN of the object.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-employ** is the DN of the DSA which was the object's original **Master Knowledge** attribute.
- **/C=US/O=XYZ/OU=Branch4/CN=dsa/CN=dsa-ops** is the DN of the DSA which is the object's new **Master Knowledge** attribute.

The administrator's selections are highlighted in bold.

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, Postal Address, Fax Number, MHS O/R Address, MHS DL Submit Permission or MHS O/R Name, 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 by using the Modify Subtree operation:

- **Presentation-Address**
- **Master-Knowledge**

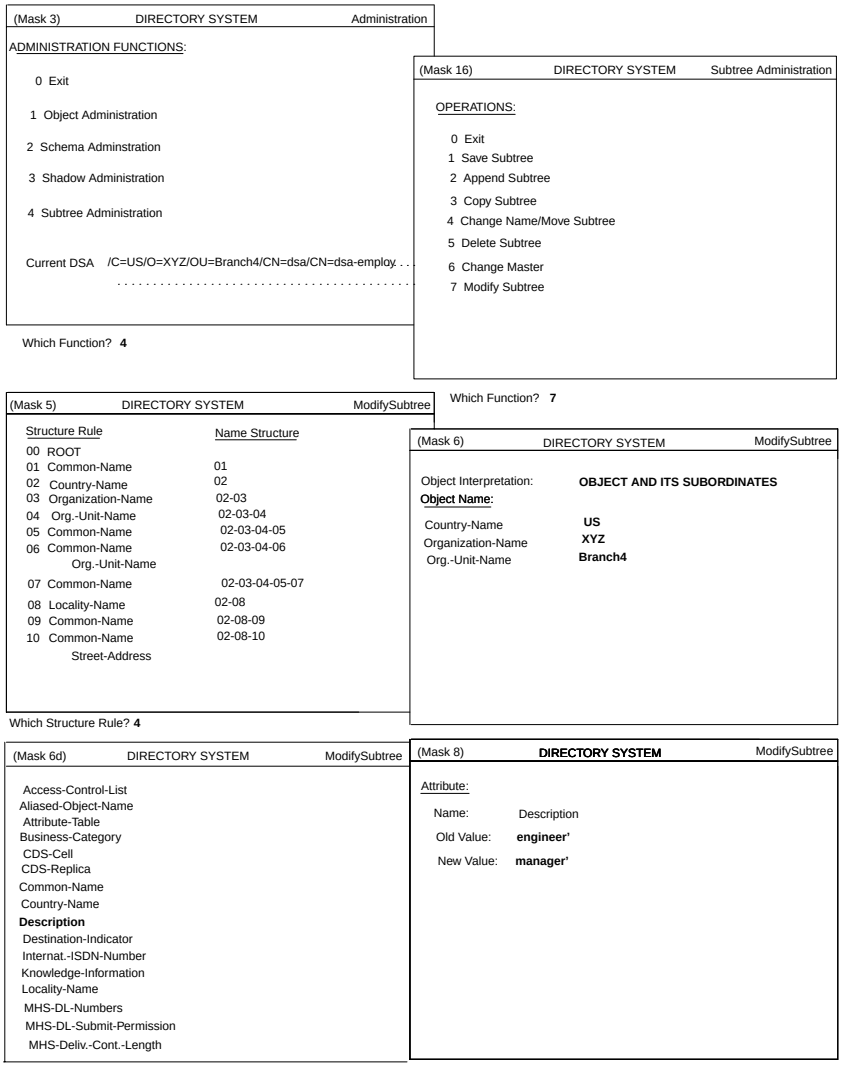
The same applies to special attributes with the following syntaxes:

- **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**. The administrator's selections are highlighted in bold.

Figure 11–28. Sample Modify Subtree Operation



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. (For more information see Chapter 9.)

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

Rule Number	Superior Rule Number	Acronyms of Naming Attribute	Acronym of Structural Object Class
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	7	CN	DNA
16	2	L	LOC
17	15	CN	REP
18	15	CN, STA	REP

Note:

- For an explanation of the acronyms of the naming attributes, see Table C-1 in Appendix C.
- For an explanation of the acronyms of object classes, see Table B-1 in Appendix B.

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

Rule Number	Superior Rule Number	Acronyms of Naming Attribute	Acronym of Structural Object Class
4	3	OU	OU
5	4	CN	ORP, APP, ORR, MDL
6	4	CN, OU	ORP
7	5	CN	DSA, APE, MMS, MTA, MUA, DNA
8	2	L	LOC
9	8	CN	REP
10	8	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. (For more information, see Chapter 9.)

Table B–1. OCT Entries

Object Class				Super-class	OID	File No.	Mandatory Attributes	Optional Attributes
Acronym	**Name**	**Kind**	**Aux.**					
TOP	Top	Abstract	—	None	85.6.0	–1	OCL	None
GTP	GDS-Top	Abstract	—	TOP	None	–1	None	ACL MK
SCH	Schema	Structural	—	GTP	43.12.2.\ 1107.1.3.6.0	0	CN	TST SRT OCT AT
ALI	Alias	Alias	—	TOP	85.6.1	–1	AON	None
C	Country	Structural	—	GTP	85.6.2	1	C	DSC SG CDC CDR

Object Class				Super-class	OID	File No.	Mandatory Attributes	Optional Attributes
Acronym	**Name**	**Kind**	**Aux.**					
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
ORP	Organizational-Person	Structural	CA	PER	85.6.7	1	None	L SPN STA MUS PDO PA PC POB FTN IIN TTI TXN X1A PDM DI RA OU TIT

Object Class				Super-class	OID	File No.	Mandatory Attributes	Optional Attributes
Acronym	**Name**	**Kind**	**Aux.**					
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
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	AM KNI PN UP
DEV	Device	Structural	—	GTP	85.6.14	2	CN	DSC L O OU SEA OWN SER CDC CDR

Object Class				Super-class	OID	File No.	Mandatory Attributes	Optional Attributes
Acronym	**Name**	**Kind**	**Aux.**					
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
DNA	DME-Nmo-Agent	Structural	—	APE	2 2.1	AA	None	
SAU	Strong-Auth.-User}	Auxiliary	—	TOP	85.6.15	–1	UC	None
CA	Certification-Authority}	Auxiliary	—	TOP	85.6.16	–1	CAC CRL ARL	CCP

For an explanation of acronyms of the mandatory attributes and the optional attributes, see Table C-1 in Appendix C.

Note: 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. (For more information, see Appendix C.)

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. (For further information, see Chapter 9.)

Table C–1. AT Entries

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	**Name**								
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

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	**Name**								
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
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	International-ISDN-Number	85.4.25	1	16	0	6	0	0	0

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	**Name**								
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
US	User-Certificate	85.4.36	1	3024	0	—	0	0	0
CAC	CA- Certificate	85.4.37	1	3024	0	—	0	0	0
ARL	Authority-Revocation-List	85.4.38	1	32503	0	—	0	0	0
CRL	Certificate-Revocation-List	85.4.39	1	32503	0	—	0	0	0
CCP	Cross-Certificate-Pair	85.4.40	1	6056	0	—	0	0	0
MDL	MHS-Deliv.-Cont.-Length	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

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	**Name**								
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-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

Attribute		OID	Lower Bound	Upper Bound	Max. No. of Val.	Syntax	Phon. Flag	Access Class	Index Level
Acr.	Name								
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
PN	Principal-Name	43.12.2.11 \ 07.1.3.4.15	1	1024	1	5	0	0	0
AM	Authentication-Mechanism	43.12.2.11 \ 07.1.3.4.16	4	4	7	9	0	0	0
AA	Alternate-Address	43.22.2.1.2.1.1	1	800	0	10	0	0	0

C.1 Explanations

The following tables provide explanations of the phonetic flags, access classes, maximum number of values, and syntaxes used in the preceding tables.

Table C–2. Phonetic Flags

Digit	Flag
0	No phonetic matching
1	Phonetic matching

Table C–3. Access Classes

Digit	Access Class
0	Public
1	Standard
2	Sensitive

Table C–4. Maximum Number of Values

Digit	Explanation
0	Unlimited number of values
>0	Upper bound for number of values

Table C–5. 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

Digit	Syntax
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
1007	Certificate
1008	Certificate Pair
1009	Certificate List
1010	Country Name
1011	Password
10000	Access Control List

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**. 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 character | (pipe) symbol.

D.1 Examples of Entering Client and Server Addresses

PSAP addresses are entered in Mask 7a of the administration program **gdsditadm**. The following figure shows how a client address is entered in Mask 7a.

Figure D–1. Example of a Client Address

```
(Mask 7a)                          DIRECTORY SYSTEM                          operation

P-Selector:
S-Selector:
T-Selector:                        Client

NSAP-Address 1:                        TCP/IP!internet=192.35.18.4+port=31010

NSAP-Address 2:                        OSILAN!ethernet=0800148101D3

NSAP-Address 3:
NSAP-Address 4:
NSAP-Address 5:
```

The following figure shows how a server address is entered in Mask 7a:

Figure D–2. Example of a Server Address

```
(Mask 7a)                          DIRECTORY SYSTEM                          operation

P-Selector:
S-Selector:
T-Selector:                        Server

NSAP-Address 1:                        TCP/IP!internet=192.35.18.2+port=31011

NSAP-Address 2:                        OSILAN!ethernet=080014151475

NSAP-Address 3:
NSAP-Address 4:
NSAP-Address 5:
```

The NSAP addresses depend on the type of the network. GDS supports TCP/IP networks using sockets and OSI-LAN/-WAN network using XTI.

GDS simultaneously operates XTI as well as sockets. If the administrator wishes to use both network types, the different addresses can be stored in the DUA cache or DSA for the same object.

If the administrator wishes 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 for this purpose.

The remainder of this Appendix provides a detailed description of all other information concerning PSAP addresses. The actual network types supported by GDS and the formats of the corresponding NSAP addresses are described in the *DCE 1.2.2 Release Notes*.

D.2 Presentation-/Session-/Transport Selector Syntax

This section deals with the presentation, session, and transport selector syntax.

```
<selector> ::=
<anystring><subset any>
OR
<anystring>'
OR
x'<hexstring>'    1)
OR
t'<anystring>'    2)
```

Note: 1) If the syntax of the frame x'<...>' is specified incorrectly, then the selector value is interpreted as **<anystring>** or **<anystring><subset any>**. 2) This indicates EBCDIC character representation of ASCII-string internally. The conversion from the internal format will be done in the following order

- ASCII
- EBCDIC
- Hex-string

```
<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)*/
<IDI-value> ::= <digitstring>
<DSP> ::=
<DSP-type>+<decimal DSP-parts>
OR
<DSP-type>+<binary DSP-parts>
```

The total length of <binary DSP-parts> must always be integral numbers of octets.

```
<DSP-type> ::=    1)
<any> /* 1 up to 8 characters */
```

Note: 1) 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.3.1).

```
<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>
OR
<slashed digitstring1>
OR
<slashed digitstring2>
<hexstrings> ::=
<hexstring>
OR
<dotted digitstring>
OR
<special ASCII-string>
OR
<digitstring>
OR
<slashed digitstring1>
```

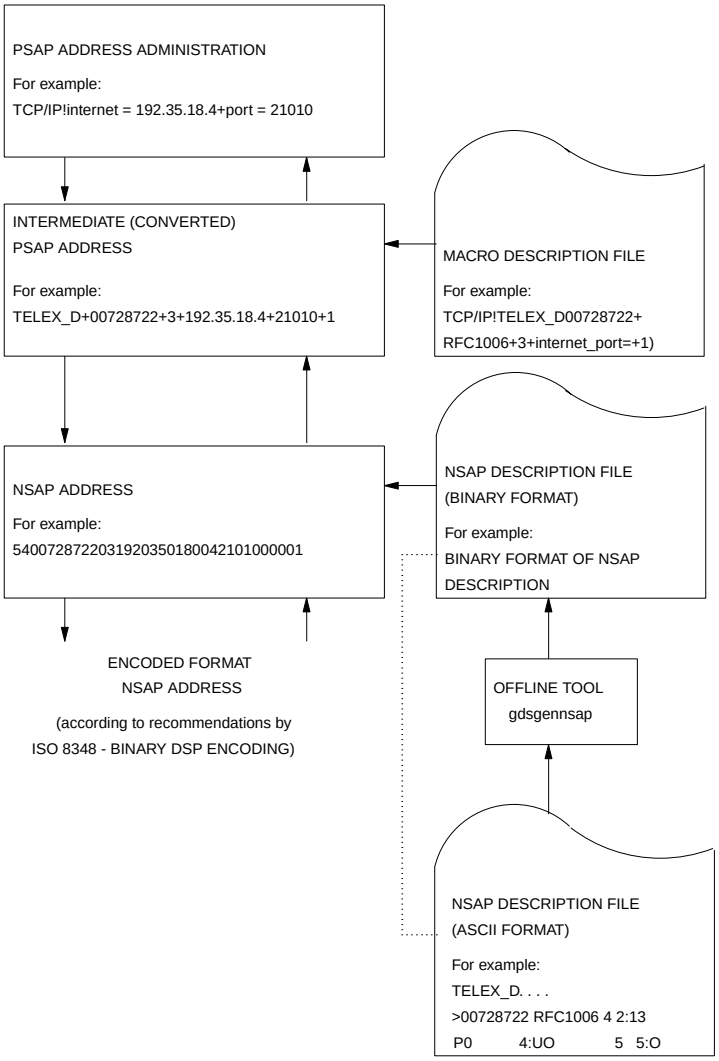
```
OR
<slashed digitstring2>
<digitstring> ::=
<digit value>
OR
<digit value><digitstring>
<dotted digitstring> ::=
.<special digitstring>
OR
<special digitstring><multi dotted digitstring>


<multi dotted digitstring> ::=
.<special digitstring>
OR
v.<special digitstring><multi dotted digitstring>
<slashed digitstring1> ::=
/<special slashed string1>
OR
<special slashed string1> <multi slashed string1>
<multi slashed string1> ::=
/<special slashed string1>
OR
/<special slashed string1> <multi slashed string1>
<slashed digitstring2> ::=
/<special slashed string2>
OR
<special slashed string2> <multi slashed string2>
<multi slashed string2> ::=
/<special slashed string2>
OR
/<special slashed string2> <multi slashed string2>
<special slashed string1> ::= [0-255]
<special slashed string2> ::= [0-65536]
<special ASCII-string> ::=
'<subset anystring0>'
<subset anystring0> ::=
<any0>
OR
<any0><subset anystring0>
```

```
<any0> ::= [printable ASCII-character set (\040 - \046,
\050 - \052, \054 - \176)]


<hexstring> ::=
<hexvalue>
OR
<hexvalue><hexstring>
<digit value> ::= [0-9]
<special digitstring> ::= [0 - 255]
<hexvalue> ::= [0-9a-fA-F]
```

Figure D–3. Common NSAP Address Syntax



D.3.1 OSI-NSAP Address Format Description Table

The following table 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. For each supported NSAP address type there can be one or more entries. Each entry consists of:

- The name of the NSAP address type (only predefined names can be used, see below) 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) (global NSAP address types only) for which the following syntaxes are permitted:

Table D–1. NSAP Address Types

SYNTAX	**MEANING**
<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 in DSP format (up to 8 characters)
- The number of DSP components (up to **MAX_DSP_PARTS**)
- The DSP part description
- A DSP part can be described by the following:
  - — DSP part size
  - — DSP part unit type

- — DSP structure (Format Identifier) identifier
- — DSP Ranges
- — DSP part of variable size
- — Optional DSP part
- — DSP part padding character
- — DSP part Justification

- DSP part size

  This indicates the size of the DSP part. The size of DSP parts is given in terms of units. The size field consists of two subfields. The first subfield, if specified, indicates the minimum number of unit values that must occur in the DSP part. The second subfield is mandatory and should be present for all DSP parts; it indicates the size of the DSP part and the maximum number of unit values that can occur for the DSP part.

- DSP part unit types:

  The unit type indicates how the conversion should take place. Following are the allowed unit types:

  1. Decimal digit
  2. Octet
  3. SemiOctet
  4. Dotted decimal string
  5. Slashed decimal string
  6. ASCII Character

  Various flags are described that can be used for specifying the DSP part unit type.

- DSP structure identifier:

  This will allow DSP format identifiers. It is possible to identify different DSP formats by having some fixed values in DSP parts. It is possible to have multiple format identifiers in a DSP format description; this allows the different DSP address formats that will occur as a result of autonomous subdomains that are allowed to make their own DSP structures. All the format identifiers should be entered complete. The length of the format identifier will be completed depending upon the maximum length of the DSP part specified in the length field, right or

left justification with the padding characters specified in the description. It is permissible to specify unit-types of decimal digits, semioctets, octets, or ASCII characters with the format identifier.

- DSP Ranges:

  Allows the user to specify the range of values that can be administered for the specific DSP part. If the format identifier flag is set then it is not permissible to specify the ranges with the specific DSP part. Each unit type type has a range by default. It is possible to further restrict the default values. In some unit types it is also possible to define the range on the ‘‘unit type strings’’.

  Following are the default values for the unit type ranges:

  If decimal digit, then lower range = 0 upper range = 9

  If semi octet, then lower range = 0 upper range = F/f If octet, then lower range = 00 upper range = FF/ff

  Note: Only hexadecimal ranges are allowed.

  For dotted decimal string: lower range = 0 upper range = 255

  For slashed decimal string: lower range = 0 upper range = 255 OR lower range = 0 upper range = 65536

  depending upon the unittype in the slashed notation. It is not permissible to specify the ranges for unittype ASCII character.

  For the following unittypes, it is possible to define the ‘‘unit type string range’’: Decimal digit Semi Octet Octet

  The default is always that ‘‘unit type range’’ and ‘‘unit type string range’’ must be specified explicitly.

- DSP parts of variable size:

  In some DSP structures the length of the DSP part is given by the prefix. Each variable DSP part, if it is not the last part of DSP part structure, must have its length as prefix to the DSP part structure. The prefix type will be indicated by a special flag.

- DSP parts which are optional:

  This means the DSP part is optional. A DSP part can be specified as optional if and only if it is the last part of the DSP structure. An optional DSP part can be variable sized or of fixed size.

- DSP part padding mechanism:

  Different characters can be used as padding characters in different formats of DSP. The padding characters will be encoded or decoded depending upon the unit type.

  In most of the unit types the padding characters can be entered as p<PADCHAR>. In case of ASCII unit types the printable padding character can be entered as normal, while nonprintable (and also printable) ASCII characters can be entered using ‘‘\’’ (backslash) followed by the decimal ASCII value of the character.

  The padding character ‘‘-’’ (dash) has special meaning, and can be specified with the parts where it is not necessary to have padding characters and input is given as a complete string.

- Leading/Trailing/Justification:

  As explained for ‘‘padding characters’’. It should be possible to say whether the DSP part is to be right justified or left justified if it is less than its size; depending upon the justification, padding characters should be added.

NOTE:

1. If the format identifier is specified then it is the responsibility of the offline software to expand the format identifier according to the description entered by the user in the format description. The expanded form of the format identifier is unique in the format description file.
2. DSP Part flags can be specified in any order.

Notation:

1. The following description of the DSP format of the OSI NSAP address is given in BNF notation. All the terminal symbols are printed as they will appear, and all the non terminal symbols are enclosed in ‘‘<’’ and ‘‘>’’ symbols. If one of the symbol ‘‘<’’ or ‘‘>’’ appears in the input as a terminal symbol, it is indicated by the ‘‘\’’ symbol; for example, ‘‘\>’’ means ‘‘>’’ as terminal symbol. Comments are given in /* ..... */ , this has nothing to do with BNF notation and should not be confused with it. Similarly, some explanations are given in square brackets (‘‘[’’ ‘‘]’’) preceded by the colon character (‘‘:’’) for the better understanding of BNF notation description.
2. To improve legibility, all the capital non-terminal symbols are expanded where possible into terminal symbols.

```
<OSI_NSAP_Description> ::= <NSAP_Format_Name>
OR
<NSAP_Format_Name><t><DSP_Desc>
<NSAP_Format_Name>     ::= <AsciiSTR>  /* Values are predefined */
<DSP_Desc>             ::=
<DSP_Format_Name><t><NUMOFPARTS><t><DSP_Parts_Desc><t>
><t><DSP_Desc>
OR
<DSP_Format_Name><t><NUMOFPARTS><t><DSP_Parts_Desc>
OR
<idi><t><DSP_Format_Name><t><NUMOFPARTS><t><DSP_Parts_Desc><t>
><t><DSP_Desc>
OR
<idi><t><DSP_Format_Name><t><NUMOFPARTS><t><DSP_Parts_Desc>
<idi>   ::= <DigitStr>  /* Depending upon size of IDI */
OR
<DigitStr>-<Digitstr>
OR
*
OR
+
<DSP_Format_name>       ::= <AsciiSTR>  /* Up to max 8 characters */
<NUMOFPARTS>            ::= [1-10]
<DSP_Parts_Desc>        ::=
<DSP-part-size>
OR
<DSP-part-size>:<DSP-format-description list>
OR
<DSP-part-size>:<DSP-format-description list><t><DSP_Parts_Desc>
<DSP-part-size>    ::=  <max.  DSP-units>
OR
<min.  DSP-units>,<max.  DSP-units>
<max.  DSP-unit size> ::= <DIGIT>
OR
<DIGIT><DIGIT>
<min.  DSP-unit size> ::= <DIGIT>
OR
<DIGIT><DIGIT>
:[Max. limit depending upon context
but in no case the expanded form of
```

```
it should exceed the max. allowed
DSP size.  Example:
"2,4"
This means a minimum of two and maximum of
four unittypes can occur ]
```

```
<DSP-format-description-list> ::=
<DSP-format-unit-description>
OR
<DSP-format-unit-description>,<DSP-format-description-list>
<DSP-format-unit-description> ::=
u<type-value> OR r<RangeValue> OR R<RangeUnitString> OR
I<format-id> OR V OR O OR v OR p<PADCHAR> OR  L
```

Meaning of terminal symbols defined for <DSP-format-description>:

u : Indicates the unit type of the DSP part.

r : The unittype range is default for units and can be further restricted.

R : Special unittype string ranges can be specified for some unit types (‘‘ud’’,‘‘uD’’,‘‘uh’’,‘‘uH’’ , the meaning of flags are explained later in the description).

I : Indicates that the specific DSP part is format identifier.

V : Indicates that the specific DSP part is variable sized.

O : Indicates that the specific DSP part is optional.

v : Indicates the prefix length is required (this flag has meaning only if the DSP part is the last part and the user has option to prefix the length)

p : Indicates the character that follows should be used as padding character.

L : Indicates the DSP part should be left justified (default is right justification and should not be specified).

```
<type-value>  ::= d OR D OR h OR H OR a OR A u OR o OR O
OR s1 OR s2 OR S1 OR S2
```

Meaning of the terminal symbols defined for <type value>

Note: There will be two levels of encoding. The first level of encoding will be done on the basis of unit flags defined. The second level of encoding will depend upon the DSP syntax. In the description of each flag a unit is defined which indicates the number of decimal digits / hex digits. Compatibility between first level and second level encoding will be taken care of by the software.

d : A unit consists of one input character. Each unit in the input will be converted into one decimal digit, and vice versa. Only decimal digits can be entered as input. Allowed for both DSP syntaxes.

Example: "3:ud": "123" ==> "123"

D : A unit consists of two input character. Each unit in the input will be converted into two decimal digits, and vice versa. Only an even number of decimal digits can be entered as input. Binary DSP syntax only.

Example: "4:uD": "12345678" ==> "12345678"

h : A unit consists of one input character. Each unit in the input will be converted into a hex digit, and vice versa. Only hex digits can be entered as input. Binary DSP syntax only.

Example: "4:uh": "ABCD" ==> "ABCD"

H : A unit consists of two input character. Each unit in the input will be converted into two hex digits, and vice versa. Only an even number of hex digits can be entered. Binary DSP syntax only.

Example: "4:uH": "AB3456EF" ==> "AB3456EF"

a : A unit consists of one ASCII character. Each unit in the input will be converted into three decimal digits, and vice versa. All printable ASCII characters can be entered. Allowed for Decimal and Binary DSP syntax.

Example: "2:ua": "EF" ==> "069070"

A : A unit consists of one ASCII character. Each unit in the input will be converted into two hex digits, and vice versa. All printable ASCII characters can be entered. Binary DSP syntax only.

Example: "2:uA": "EF" ==> "4546"

o : A unit consists of one dotted string. Each unit in the input will be expanded into three decimal digits, and vice versa. Allowed for Decimal and Binary DSP syntax.

Example: "4:uo" : "127.0.0.1" ==>"127000000001"

O : A unit consists of one dotted string. Each unit in the input will be converted into two hex digits, and vice versa. Binary DSP syntax only.

Example: "4:uO" : "127.0.0.1" ==>"7F000001"

s1 : A unit consists of one slashed decimal string. Each unit will be expanded into three decimal digits, and vice versa. Allowed for Decimal and Binary DSP syntax.

Example: "2:us1" : "25/155" ==>"025155"

s2 : A unit consists of one slashed decimal string Each unit will be expanded into five decimal digits, and vice versa. Allowed for Decimal and Binary DSP syntax.

Example: "2:us2" : "425/1631" ==>"0042501631"

S1 : A unit consists of one slashed decimal string. Each unit will be converted into two hex digits, and vice versa. Binary DSP syntax only.

Example: "2:uS1" : "25/155" ==>"199B"

S2 : A unit consists of one slashed decimal string. Each unit will be converted into four hex digits, and vice versa. Binary DSP syntax only.

Example: "2:uS2" : "425/1631" ==>"01A9065F"

```
<RangeValue>    ::= <DigitStr>-<DigitStr> OR <HexStr>-<HexStr>
For example :
4:uh,r0-FF    ( range for hexadecimal notaion)
4:uo,r0-125   (unit type range for dotted notat.)
: [ if the unit type is "uo" or "uO" or "us1" or
"us2" or "uS1" or "uS2" then only unit type ranges
will be allowed; that is:
For "uo" or "uO" ranges between 0-255 are allowed
For "us1" or "uS1" ranges between 0-255 are
allowed
For "us2" or "uS2" ranges between 0-65536 are
```

```
allowed. The defaults are the minimum and maximum
allowed in the above specification ]
<RangeUnitString>  ::= <DigitStr> /* Constant or Reserved Values */
OR
<DigitStr>-<DigitStr>
OR
<HexStr> /* Constant Value or Reserved Values */
OR
<HexStr>-<HexStr>
For example : 4:ud,R1234-5432
<format-id>     ::= <string>
For example :
"1:uD,I08" ==> "08"
"1:ua,I8"  ==> "056"
"1:uo,I8"  ==>  [ NOT ALLOWED ]
<String>     ::= <DigitStr> OR <HexStr> OR <AsciiStr>
<PADCHAR>    ::= [0-9,'-']   /* if unit flag is 'd' or 'h' or
'o' or 's1' or 's2' */
::= [0-9A-Fa-f,'-'] /* if unit flag  'D' or 'H' or
'O' or 'S1' or 'S2' */
::= [ 00-127] /* if unit flag is 'a' or 'A' */
:    [ It is possible to produce all printable characters using the
above values. Nonprintable as well as printable characters can
be produced with backslash notation. In backslash notation the
character can be input as a decimal ASCII value. For example:
"4:ud,p0" : "12" ==> "0012"
"4:uH,p0" : "EF" ==> "000000EF"
"4:uA,p0" : "EF" ==> "30304546"
"4:uA,p\048: "EF" ==> "30304546"
"4:uA,p\0" : "EF" ==> "00004546"
"4:uA,p0" : "0F" ( Not allowed ) ]
```

Note: The character ‘‘-’’ has special meaning used in conjunction with the padding character flag. This character indicates that the field should be input complete; no padding will be done by software.

```
<DigitStr>      ::= <DIGIT>    OR <DIGIT><DigitStr>
<HexStr>        ::= <HEXDIGIT> OR <HEXDIGIT><HexStr>
<DIGIT>         ::= [0-9]
```

```
<HEXDIGIT>      ::= [0-9A-Fa-f]
<AsciiStr>      ::= '<AsciiSTR>'
<AsciiSTR>      ::= <CHARACTER> OR <CHARACTER><AsciiSTR>
<CHARACTER>     ::= /* Any printable Character (0x20-0x7E) */
<t> ::=  space(s)or tab character.
```

Default values for DSP format Description:

- DSP Part size is fixed.
- DSP Part is mandatory and should always be entered.
- All values allowed for a particular DSP syntax can be entered; that is, no ranges and no format identifier.
- Unit Type is ‘‘ud’’ if DSP syntax is decimal; ‘‘uH’’ if DSP syntax is binary.
- DSP Part is right justified.
- Padding character is ‘‘0’’.

Following are the contents of the NSAP Description file that comes with the standard software. They can be customized, if required, to conform with local environments.

NOTE:

1. Lines beginning with a # character are treated as comments.
2. In the description of DSP structures ‘‘+’’ signs are used as separators.

```
# -- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- -----
# 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
# The DIN has defined the DSP Structure as follows :
#             DE_FK   ( DE Bereichskennung )
#                |      DE_BK ( DE FormatKennung )
#                |      |         FI  ( Format Identifier )
#                |      |         |        RI ( Regional Identifier )
#                |      |         |        | Reserved
#                |      |         |        |   |   RoutingDomain
#                |      |         |        |   |      |  Area
#                |      |         |        |   |      |  | System
#                |      |         |        |   |      |  |  |  N-Sel
#                |      |         |        |   |      |  |  |  |
DCC_B 276  DIN   9 1:I3,uh 3:I100,uh 1:I01 1 2:R0000  2  2  6  1
#
# FORMAT=CCITT-F.69;AFI=40,54;IDI-length=8;DSP-syntax=decimal;
#    DSP-length=30
#
#                           Prefix
#                              |     Type
#                              |      |        DTE
#                              |      |         |
TELEX_D 00728722 INTX25_1 3  2:I1   1:I0       26
#
#                            Prefix
#                              |      Type
#                              |       |       PID
#                              |       |        |      DTE
#                              |       |        |       |
> 00728722  INTX25_2  4      2:I1   1:I1     4:ua,V  14:ud
#
#                             Prefix
#                               |      Type
#                               |        |       CUDF
#                               |        |        |      DTE
#                               |        |        |       |
>     00728722  INTX25_3  4 2:I1   1:I2     4:ua,V  14:ud
# As Described above for INTX25
>         00728722  JANET_1   3     2:I2   1:I0      26
>         00728722  JANET_2   4     2:I2   1:I1      4:ua,V  14:ud
>         00728722  JANET_3   4     2:I2   1:I2      4:ua,V  14:ud
```

```
#
#                              Prefix
#                               |      IP Address
#                               |          |      Port
#                               |          |       | Transport-Set
#                               |          |       |     |
>     00728722 RFC1006  3   2:I3,p0     4:uo    5     5:O
#
# FORMAT=CCITT-F.69;AFI=41,55;IDI-length=8;DSP-syntax=binary;
#    DSP-length=15
#
#                              NetBios
#                               |  Subnet-Id
#                               |     |    MachineName
#                               |     |          |
TELEX_B  50093994  NETBIOS   2     2          8
#
# FORMAT=CCITT-E.163;AFI=42,56;IDI-length=12;DSP-syntax=decimal;
#    DSP-length=26
#
PSTN_D
#
# FORMAT=CCITT-E.163;AFI=43,57;IDI-length=12;DSP-syntax=binary;
#    DSP-length=13
#
PSTN_B
#
# 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
#
#                   DFI  (DSP Format Identifier)
#                    |       SFI  (System Format Identifier)
#                    |        |     Country
#                    |        |       |   Reserved
#                    |        |       |      | RoutingDomain
#                    |        |       |      |     |  Area
#                    |        |       |      |     |   |System
#                    |        |       |      |     |   |   |  N-Sel
#                    |        |       |      |     |   |   |    |
ICD_B 0058 OSILAN 8 1:I80    1:I01  2:uA  2:R0000  2   2  6:uD  1
>        NEA    8 1:I80    1:I02  2:uA  2:R0000  2:uD  2:uD  6:uD,p0 1
>        TCP/IP 8 1:I80    1:I05  2:uA  2:R0000  2   2  4:uo  1
>        X.25   8 1:I80    1:I06  2:uA  2:R0000  2   2  6     1
>        ISDN   8 1:I80    1:I07  2:uA  2:R0000  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
#
# 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 table is contained in the ASCII file **nsapform**. The communications software uses a binary representation of this table contained in the file **nsapform.cfg**. For this reason, each time a change is made in the ASCII file, it must be reconverted

into its binary version **nsapform.cfg** by means of the tool **gdsgennsap**, which is invoked as follows:

% gdsgennsap *ASCII_file_name binary_file_name*

Pertinent file locations are as follows:

- **/opt/dcelocal/var/adm/directory/gds/adm/nsapform**
- **/opt/dcelocal/var/adm/directory/gds/conf/nsapform.cfg**

D.3.2 Concrete DSP Syntaxes

This section describes the concrete DSP syntax.

```
<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 RFC1006>::=
RFC1006+<prefix>+<internet-address>+
<port-number>+<transport-set>
<binary NETBIOS-DSP>::=
NETBIOS+<prefix>+<subnet-id/B>+<host name/B>
+<unique/group-id>
<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>
<host name/B> ::= '<special ASCII-string>'
<unique/group -id> ::= <digitstring>
```

D.4 Macro Facility

The facility for defining macros is supported in order to simplify the task of entering NSAP addresses for the user. There are two different types of macro; the first must always be defined in the form <*macro name*>=<*macro value*> and the second must always be defined in the form <*macroname*>*macro value/macro parameter list*> (see the following text). Macros are recognized by the administration program in both directions (screen input/output). Thus, the semantics of the two macro types is 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 that matches <*macro name*>= is substituted by the value <*macro value*> of the macro.

- Writing NSAP information to the screen

  The specified output string is parsed, and every (sub)string that matches <*macro value*> is substituted by the name <*macro name*>= of this macro.

  Note: Macros of this type are substituted recursively:

```
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 this macro <*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 in this case, macros of this type need to be defined definitively.

  Output strings are first parsed for macros of type 2. They are only parsed for macros of type 1 if there is no matching.

  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 is not recommended because the results are unpredictable. One macro type is not to be used within a macro of the other type.

D.4.1 NSAP Address Macro Definition Syntax

This section describes 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
<subset anystring2>
OR
<subset anystring2>+<macro parameter
list definition>


<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 describes 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 with no DSP:

```
X121_D+12345678901234
```

- **CCITT-X.121** address with **ECMA-117** structured binary DSP:

```
X121_B+12345678901234+ECMA117+01+08227619+FE
```

- **CCITT-F.69** address with **RFC1006** structured decimal DSP:

```
TELEX_D+00728722+RFC1006+3+127000000001+4711+1
```

D.5.1 Examples of Macro Definitions

Following are examples of macro definitions.

```
TCP/IP!TELEX_D+00728722+RFC1006+3+internet=+port=+1
INTX25!TELEX_D+00521090+INTX25+1+dte=+pid=
LAN2!LOCAL_B+ECMA117+1+ethernet=+00
WAN-CONS!X121_B+dte=
ISDN-CONS!ISDN_B+isdn=
# The following NSAP macros are for SNI/Siemens internal use only
ETH-CLNS!ICD_B+58+OSILAN+01+01+'DE'+0000+00+01+ethernet=+FE
WAN-NEA!ICD_B+58+NEA+01+02+'DE'+0000+00+region=+processor=+00
TCP-IP!ICD_B+58+RFC1006+01+05+'DE'+0000+00+01+internet=+00
NETBIOS!TELEX_B+50093994+NETBIOS+01+1+hostname=+u/gid=
TCP/IP!TELEX_D+00728722+RFC1006+3+internet=+port=+1
NETBIOS!TELEX_B+50093994+NETBIOS+01+1 +hostname=u/gid=
X25=X121_D+1234567890
```

D.5.2 Examples of Macro Calls

The input strings are expanded as follows, provided that the macros defined in Section D.5.1 are present.

The following macro:

```
X25=1234
```

is expanded into:

```
X121_D+12345678901234
```

The following macro:

```
TCP/IP!internet=127.0.0.1+port=4711
```

is expanded into:

```
TELEX_D+00728722+RFC1006+3+12700000001+4711+1
```

The following macro:

```
NETBIOS!hostname='machine1'+u/gid=01
```

is expanded into:

```
TELEX_B+50093994+NETBIOS+01+1+'machine1'+01
```

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	BYELORUSSIAN SSR	BY
CAMEROON	CM	CANADA	CA
CAPE VERDE	CV	CAYMAN ISLANDS	KY

Country Name	Code	Country Name	Code
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
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 TERRITORIS	TF	GABON	GA
GAMBIA	GM	GERMAN DEMOCRATIC REPUBLIC	DD

Country Name	Code	Country Name	Code
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

Country Name	Code	Country Name	Code
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
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 REPLUBLIC OF	TZ

Country Name	Code	Country Name	Code
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
UKRAINIAN SSR	UA	UNITED ARAB EMIRATES	AE
UNITED KINGDOM	GB	UNITED STATES	US
UNITED STATES MINOR OUTLYING ISLANDS	UM	URUGUAY	UY
USSR	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		

Notes to Table:

[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

The following table 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.

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			¡	±	`		Æ	æ
2			”	2	B	R	b	r			¢	²	´			đ
3			1)	3	C	S	c	s			¶	³	^		a	
4			1)	4	D	T	d	t			$	x	~		Ħ	ħ
5			%	5	E	U	e	u				µ	¯			
6			&	6	F	V	f	v			#	⊥			IJ	ij
7			’	7	G	W	g	w			§	•			L·	l·
8			(	8	H	X	h	x				÷				
9			)	9	I	Y	i	y								
A			*	:	J	Z	j	z							Œ	œ
B			+	;	K	[	k				<<	>>				ß
C			,	<	L		l	|				¼	_			
D			–	=	M	]	m					½	”		Ŧ	ŧ
E				>	N		n					¾			η	η
F			/	?	O	_	o								’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.

When the GDS administration programs attempt to convert following characters from the T.61 codeset to the ISO 8859-1 codeset, a ? (question mark) is displayed:

- e0 (Ω)
- e4 (H with stroke)
- e6 (|J)
- e7 (L with middle dot)
- e8 (L with stroke)
- ea (OE)
- ed (T with stroke)
- ee (η)
- ef (η)
- f0 (K)
- f2 (ð)
- f3 (eth)
- f4 (h with stroke)
- f5 (i without dot)
- f6 (ij)
- f7 (l•)
- f8 (l with stroke)
- fa (œ)
- fd (t with stroke)
- fe (η)

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	á	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	^	0xc3	a, A, c, C, e, E, g, G, h, H, i, I, j, J, o, O, s, S,
accent			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.

Bold characters cannot be reverse mapped.

All characters that standalone must be followed by a space.

If the following diacritical characters are followed by a space, they cannot be mapped to the 8859-1 codeset:

- breve
- dot above
- umlaut
- ring
- double accent
- ogonek
- caron

The following table shows the invalid combinations of diacritical letters when converting from the T.61 codeset to the ISO 8859-1 codeset. The GDS administration programs displays a ? (question mark) when it encounters a combination that is not defined in the ISO 8859-1 codeset.

Table E–5. Invalid ISO 8859-1 Combinations of Diacritical Characters and Basic Letters

Name	**Repr.**	**Code**	**Valid Basic Letters Following**
grave accent	`	0xc1	
acute accent	á	0xc2	c, C, g, l, L, n, N, r, R, s, S, z, Z
circumflex	^	0xc3	c, C, g, G, h, H, j, J, s, S, w, W, y, Y
accent			
tilde	~	0xc4	i, I, u, U

Name	Repr.	Code	Valid Basic Letters Following
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	Y
ring	°	0xca	u, U
cedilla	¸	0xcb	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

E.3 ISO 8859-1 (Latin-1) Syntax

The following table shows the valid set of ISO 8859-1 characters. (The row headings indicate the lower four bits and the column headings show the higher four bits of the encoding in hexadecimal.)

Table E–6. ISO 8859-1 (Latin-1) Code Set

	2	3	4	5	6	7	A	B	C	D	E	F
0	SP	0	@	P	‘	p	NBSP	°	À	Đ	à	ð
1	!	1	A	Q	a	q	¡	±	Á	Ñ	á	ñ
2	"	2	B	R	b	r	¢	2	Â	Ò	â	ò

3	#	3	C	S	c	s	£	3	Ã	Ó	ã	ó
4	$	4	D	T	d	t	¤	´	Ä	Ô	ä	ô
5	%	5	E	U	e	u	¥	μ	Å	Õ	ą	õ
6	&	6	F	V	f	v	|	¶	Æ	Ö	æ	ö
7	’	7	G	W	g	w	§	•	Ç	×	ç	÷
8	(	8	H	X	h	x	¨	¸	È	Ø	è	ø
9	)	9	I	Y	i	y	©	1	É	Ù	é	ù
A	*	:	J	Z	j	z	ª	º	Ê	Ú	ê	ú
B	+	;	K	[	k	{	<<	>>	Ë	Û	ë	û
C	,	<	L	\	l	|	¬	$\frac{1}{4}$	Ì	Ü	ì	ü
D	-	=	M	]	m	}	SHY	$\frac{1}{2}$	Í	(aaY	í	
E	.	>	N	^	n	~	®	$\frac{3}{4}$	Î	þ	î	þ
F	/	?	O		o	DEL	¯	¿	Ï	ß	ï	ÿ

When the GDS administration programs attempt to convert any of the characters shown below from the ISO 8859-1 codeset to the T.61 codeset, the following error message is displayed:

```
ERROR: ONE OF THE FIELDS COULD NOT BE CONVERTED
TO THE T.61 STRING
```

The ISO 8859-1 codeset characters that will cause this to happen are:

- 5c (\)
- 7b ({)
- 7d (})
- a6 (|)
- a9 (©)
- ac (¬)
- ad (SHY)

- ae (®)
- b9 (^{1})

Appendix F

Worksheets

This appendix contains seven sample GDS configuration worksheets that administrators can copy for their own use. Administrators 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: _____

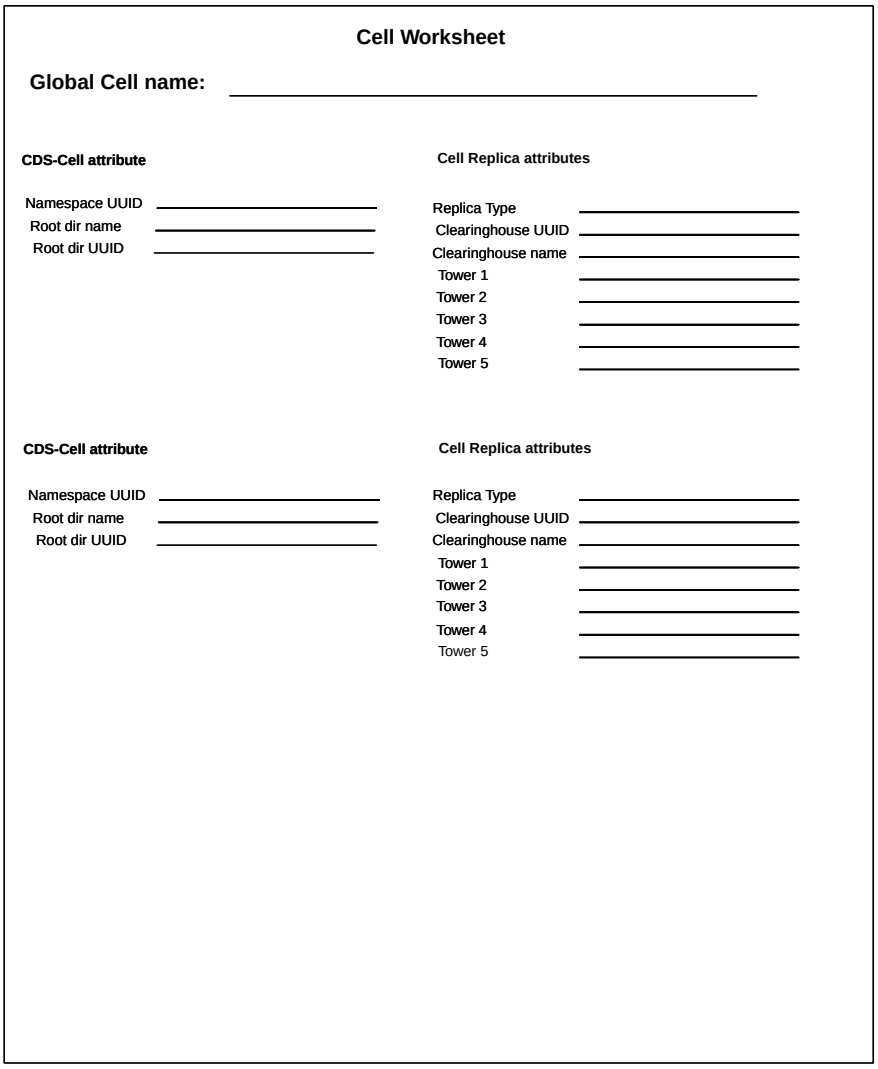
Access Class	**DN of User**	**Interpretation** (Single object or subtree)

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

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, 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

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)

Client/Server Worksheet

Global Cell Name:_____

Name of client/server machine: _____

Distinguished name of DSA: _____

PSAP address:

NSAP address _____

Transport procotol _____

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

DUA Cache Information

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Global Cell Name:_____

Name of client/serve machine: _____

Distinguished name of DSA: _____

PSAP address:

NSAP address _____

Transport procotol _____

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

DUA Cache Information

Distinguished name of DSA	DSA-Type	General DSA type (remote GDS, remote non-GDS, intial, first-level)
_____	_____	_____
_____	_____	_____
_____	_____	_____

GDS Remote and Non-GDS DSA Worksheet

Distinguished name of DSA: _____ DSA type _____

PSAP address (DSA)

NSAP address _____

Transport protocol _____

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

Distinguished name of DSA: _____ DSA type _____

PSAP address (DSA)

NSAP address _____

Transport protocol _____

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

Distinguished name of DSA: _____ DSA type _____

PSAP address (DSA)

NSAP address _____

Transport protocol _____

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

Distinguished name of DSA: _____ DSA type _____

PSAP address (DSA)

NSAP address _____

Transport protocol _____

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

Shadow Update Worksheet

Initiator: _____

Target (shadow DSA): _____

Distinguished Names of Entries	**Interpretation** (object/subtree)	**Update Frequency** (HIGH, LOW, or MEDIUM)	**Update Time** (minutes, hours, or days)

Appendix G

ASN.1 Representations

Table G-1 describes the ASN.1 representations that are used by **gdscstub** and **gdsdsa** for ASN.1 encoding and decoding. The values used by **gdsdsa** are stored in the following file (*x* is the directory ID used by **gdssysadm**):

/opt/dcelocal/var/directory/gds/dsa/dir*x* **/asn1_attr**

The values used by **gdscstub** are stored in the following file:

/opt/dcelocal/var/adm/directory/gds/conf/asn1_attr

Table G–1. ASN.1 Representations

ASN.1 Representation	Explanation
1	Distinguished Name
2	Object Identifier
3	Case Exact String
4	Case Ignore String

ASN.1 Representation	Explanation
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	TTX ID
1004	Fax Number
1005	Preferred Delivery Method
1006	Presentation Address
1010	Country Name
1011	Password
10000	Access Control List
10002	T.61 String
10003	ASN.1 String
10004	Integer List

ASN.1 Representation	Explanation
10005	Printable String List
10006	T.61 String List

Appendix H
The Network Directory Service (NDS)

H.1 Introduction

Communications networks all require a method of addressing users of the network. In data communications, users of the network are addressed through the systems that are attached to the network. In OSI networks, systems that are attached to the network are identified by globally unique NSAP addresses. The basic formats of such NSAP addresses are described in detail in the framework document ISO 8348/AD2.

This chapter now describes the NDS, a service which provides the mapping of such global NSAP addresses into local address information prescribed in syntax and semantic by the different transport service interfaces (such as XTI, socket), transport service providers (for example, TCP/IP, OSI-LAN (ethernet, tokenring), OSI-WAN (X.25, ISDN)) and vice versa.

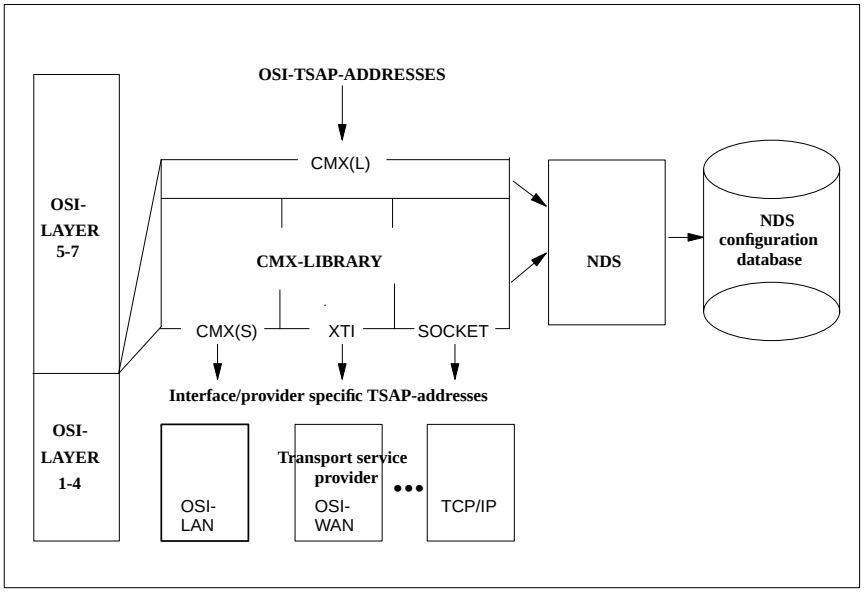
The semantic of the local address information may be equal to the global NSAP address information in some cases (if the NSAP address formats used contain the full NSAP address information) or it may be partially or totally different (if the NSAP address formats used contain only some or no address information). Furthermore,

the mapping of NSAP addresses must be done under consideration of additional information which may be dependent on the system configuration and the location of the system.

To fulfill all the requirements and to make the mapping of addresses as flexible as possible a configuration database is used by the NDS. This database is created by the NDS compiler during compilation of a NDS configuration source file. The NDS configuration database is contained in the file **NDSCONF.DIB**, and the default NDS configuration source file in the file **NDSCONF.dat**. The configuration source file **NDSCONF.dat**, which is provided with GDS contains information about all transport interfaces and providers supported by NDS. This file may be changed and customized by a system administrator if required. In the remainder of this chapter the format of the configuration source file and the use of the NDS compiler are explained in detail.

In the context of GDS the NDS is mainly used by the C-Stub and S-Stub. But any other OSI application may become a user of this service if using global PSAP addresses from a X.500 directory service. The following figure gives an overview of how the NDS is embedded in the communications architecture.

Figure H–1. NDS in the Communications Architecture



H.2 NDS configuration source file

The NDS configuration source file is a special text file from which the NDS compiler generates the NDS configuration database valid for one or more computer systems. A source file contains descriptions of:

- All transport interfaces available in a system
- All transport service providers (profiles) available in a system
- All NSAP addresses (or NSAP address formats) available and used in the network(s). Assigned to each NSAP address (format) there must be information (mapping options) describing how the NDS has to provide the address mapping.

A configuration source file is hierarchically organized to minimize the amount of data a system administrator has to enter as follows:

```
BEGIN SECTION INTERFACES
BEGIN INTERFACE <A>
        # common interface option area
<common interface specific information (options)>
BEGIN PROFILE <A.1>
         # common  profile option area
<common profile specific information (options)>
<NSAP address (format) A.1.1 description>
# NSAP address mapping option area
<NSAP address A.1.1 mapping option(s)>
. . .
<NSAP address (format) A.1.n description>
# NSAP address mapping option area
<NSAP address A.1.n mapping option(s)>
END PROFILE <A.1>
. . .
BEGIN PROFILE <A.m>
. . .
END PROFILE <A.m>
END INTERFACE <A>
. . .
BEGIN INTERFACE <Z>
. . .
```

```
END INTERFACE <Z>
   END SECTION INTERFACES
```

H.2.1 General Notes

- A configuration source file must contain at least one interface description, one profile description per interface description, and one NSAP address (format) description per profile description.
- The order of interface descriptions, of profile descriptions inside an interface description and the order of NSAP address (format) mapping options inside a NSAP address description is predefined and must not be changed (the default configuration source file **NDSCONF.dat** may be used as a reference to ensure this order).
- If a transport interface or a transport service provider (profile) is not supported on a specific system, then the corresponding description may be removed from the source file, but if required at a later time, the description must be inserted at the same location.
- Information (in particularly mapping options) specified at a higher level are transmitted to lower levels. Information specified at a lower level overrides information of the same type specified at a higher level (such as an E.163 country code option specified in the common section of an interface description is transmitted to all profile descriptions resp. NSAP address descriptions being within this interface description; an E.163 country code option specified within a NSAP address description overrides this E.163 country code option from the higher level).

The general syntax for an information statement entered into the source file is always:

< **keyword**> = <value> ;

There may be one or several statements per line. For clarity it is however advisable to enter only one statement per line.

Lines starting with a number sign (‘‘#’’) are treated as comment lines. Comments may normally occur everywhere in the source file, but not inside of statements.

The transport interfaces and transport providers (profiles) supported by the NDS are summarized in Table H-1.

Table H–1. Transport Interfaces

Transport Interface	**Transport Provider (profile)**	**Comment**
SOCKET	RFC1006	TCP/IP over RFC1006
XTI	LOOPSBKA	local loopback
	OSITYPE	ETHN-CLNS-active
	LANSBKA	ETHN-CLNS-inactive
		TR-LLC (tokenring)
	WANSBKA	WAN-CONS
		ISDN-CONS
	WAN3SBKA	WAN-X25, ISDN-X25
		ISDN-HDLC
	WANNEA	WAN-NEA, WAN-NX25
		ISDN-NEA, ISDN-NX25
	NETBIOS	TCP/IP over
		RFC1001/1002
CMX(s)	analog to XTI, excluding NETBIOS	*)

*) Because the kernel components of CMX do not consider the NDS, the CMX(s) interface is not yet used.

H.3 Elements of the NDS configuration source file

In the following all elements (keywords) of the configuration source file are described:

CONFIGURATION_FILE_TITLE Assigns an identification to the NDS configuration database. The UNIX **what** command may be applied to the database file **NDSCONF.DIB** to identify the database actually in use.

VALUE(s): 1 – 32 ASCII characters (excluding quotation mark (")).

USE: Mandatory

BEGIN *xname*, END *xname* Marks the beginning and end of a section (x = SECTION), an interface or a profile description (x = **INTERFACE** or **PROFILE**). Interface and profile names are predefined.

VALUE(s): Interface names = keywords **socket**, **xti**, **cmx**(s); profile names = following keywords:

- **RFC1006**
- **OSI-TYPE**
- **LANSBKA**
- **WANSBKA**
- **WAN3SBKA**
- **WANNEA**
- **LOOPSBKA**
- **NETBIOS**

USE: Mandatory

PROFILE_PRIORITY Assigns a priority to a profile. This priority is used by the NDS for profile selection if the provided NSAP address(es) allow(s) a connection to be established over different transport providers (profiles) (such as over a LAN network and over a WAN network). A higher value means higher priority. The assignment of the profile priority must be unambiguous (interface independent) for all profiles specified in the configuration source file.

VALUE(s): 1 – 16

USE: Mandatory

PROFILE_DEVICE Specifies the name of a XTI transport service provider. A profile device statement may occur in the common area of the XTI interface description and become valid for all profiles below XTI) and/or in the common area of any XTI

provider (profile) description and become valid for this profile only and override the device statement in the common area of the interface (if present).

VALUE(s): ASCII characters (excluding semicolon ‘‘;’’ and space ‘‘ ’’)

USE: Mandatory (for XTI transport providers only)

NSAP_ADDRESS NSAP_ADDRESS_GROUP Specifies a single NSAP address or a group of NSAP addresses. Single addresses are entered by using the same notation defined for the NSAP address part(s) of the PSAP address administration in GDS (normally a NSAP macro format — such as TCP/IP!ethernet= 127.0.0.1+port=21020). Groups of NSAP addresses are described by means of regular expressions. The syntax and semantic of such regular expressions follow the rules defined for **regcmp(3G)** in the UNIX Programmer’s Reference manual. An NSAP address specification may occur in one profile description only, but assignment to different profiles is also allowed. Note: If a transport profile description contains NSAP address group statements and also statements for single NSAP addresses which are still described by the group statement, then the single NSAP addresses should be specified first to avoid conflicts. Because NSAP addresses are stored by the NDS compiler in the configuration database in the same order they are specified in the source file, the NDS, searching for a matching NSAP address during address mapping, will never find such a single NSAP address specified after a NSAP address group. An NSAP address, provided to the NDS, always matches the regular expression of the NSAP address group specification.

VALUE(s): NSAP macro notation as defined for PSAP address administration in GDS or regular expression syntax defined in **regcmp (3G)**.

USE Mandatory

INTERNET_ADDRESS Specifies how the NDS has to obtain the internet address required at the socket interface. (This is the NSAP address mapping option for TCP/IP over **RFC1006** profile only.) The internet address may be entered explicitly in a dotted notation (such as 138.5.0.39) or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the internet address implicitly from the DSP part *no.* (such as DSP part 2) of the selected remote NSAP address.

VALUE(s): Internet address in dotted notation or keyword **IMPLICIT_DSP** *no.*.

USE Mandatory (for TCP/IP over **RFC1006** profile only).

PORTSpecifies how the NDS has to obtain the port number required at the socket interface. (This is the NSAP address mapping option for TCP/IP over **RFC1006** profile only.) The port number may be entered explicitly or the keyword **IMPLICIT_DSP**

no. may be used to cause the NDS to extract the port number implicitly from the DSP part *no.* of the selected remote NSAP address. If the port number is entered explicitly, then the port number statement must be immediately followed by a T-selector statement (see below). A list of port number/T-selector statement pairs may also occur in this case. This allows the mapping of port numbers in dependence of the T-selector provided in the actual TSAP address. The default port number 102 (as specified in **RFC1006**) is used if no port number statement is available or if no T-selector is contained in the local/remote TSAP address provided to the NDS.

VALUE(s): 1 – 32768 or keyword **IMPLICIT_DSP** *no.*.

USE: Optional

T_SELECTOR Specifies how the NDS has to obtain the port number required at the socket interface (see port number statement above). (This is the NSAP address mapping option for TCP/IP over **RFC1006** profile only.) The T-selector may be entered as an ASCII string (such as **SERVER**) or in hexadecimal notation (**x'** *hex values***'**, for example `x'0102'`).

VALUE(s): ASCII string (up to 32 characters) (excluding semicolon ‘‘;’’ and space ‘‘ ’’)or hexadecimal string (up to 32 octets).

USE: Mandatory/optional (see port number statement above).

X121_NUMBER X121_INTERNATIONAL_PREFIX X121_DATA_COUNTRY_CODE X121_DATA_NETWORK_ID_CODE Specifies how the NDS has to obtain the complete X.121 address in public and/or private X.25 networks according to CCITT recommendation X.121 (numbering plan for the international data network service). (This is the NSAP address mapping options for X.25 networks; **WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only.) The value of the **X121_NUMBER** option may be an explicit X.121 address (consisting of a NN (national number) or NTN (network terminal number), a DCC/DNIC (see below) and an international prefix (if required)) or one of the keywords **IMPLICIT_IDI** or **IMPLICIT_DSP** *no.1* [**:***no.2*] may be used to cause the NDS to extract the X.121 address implicitly from the IDI (initial domain identifier) or from the DSP part *no.1* (or from the concatenation of the DSP parts *no.1* and *no.2*)of the remote NSAP address.

VALUE(s): 1 – 15 digits or one of the keywords **IMPLICIT_IDI** or **IMPLICIT_DSP** *no.1*[**:** *no.2*].

USE: Mandatory ((X.25 networks) **WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only).

The value of the **X121_INTERNATIONAL_PREFIX** option depends from the location (country) of the end system establishing a X.25 connection. The NDS adds the value as a prefix to the X.121 address everytime an international call is made. The NDS ignores the international prefix facility (this may often be desirable in private X.25 networks), if a dash ‘‘-’’ is assigned as a prefix value or if a X.121 address is specified explicitly by the **X121_NUMBER** option.

VALUE(s): 0-9 or dash (‘‘-’’)

USE: Mandatory (only if the X.121 address is not specified explicitly).

The value of the **X121_DATA_COUNTRY_CODE** option, or alternatively of the **X121_DATA_NETWORK_ID_CODE** option, may be the 3-digit DCC (data country code) of the country (such as 262 for Germany) in which the end system establishing a X.25 connection is attached to the public X.25 network (the list of all valid DCCs is contained in annex B of recommendation X.121) or the 4-digit DNIC (data network identification code) of the X.25 network to which the end system is attached. In most cases the first 3 digits of a DNIC are equal to the DCC of the country where the end system resides. The NDS removes the DCC/DNIC value from the called X.121 address if a X.25 connection is established within the same country/X.25 network. The NDS ignores the DCC/DNIC facility (this may often be desirable in private X.25 networks), if a dash (‘‘-’’) is assigned as a DCC/DNIC value or if a X.121 address is specified explicitly by the **X121_NUMBER** option.

VALUE(s): 3-digit DCC (000-999) or 4-digit DNIC (0000-9999) or dash (‘‘-’’). Note: The DCC/DNICs defined in recommendation X.121 do not occupy all values from the reserved range. Because the NDS compiler checks for this range only, the administrator has to guarantee the validity of any DCC/DNIC entered.

USE: Mandatory (only if the X.121 address is not specified explicitly). The X.121 options international prefix and DCC/DNIC may occur in the common options area of the XTI/CMX interface or in the mapping options area of NSAP addresses (within X.25 network relevant profiles only), whereas the X.121 address option may occur in NSAP address mapping option areas only. The X.121 options may also be used in conjunction with the E.164 option in the case of a X.31 (X.25 via ISDN) communication profile or with the X.21 option in the case of a X.32 (X.25 via X.21) communication profile. At this, the specification of the X.121 options must be always done after the specification of the E.164 (X.21)option(s).

PVC_CHANNEL Specifies how the NDS has to obtain the channel number necessary to establish a permanent virtual connection in a X.25 network. (The NSAP address mapping option for X.25 networks; **WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only.) The channel number may be entered explicitly or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the channel number from the DSP part *no.* of the selected remote NSAP address.

VALUE(s): 1 – 4096 or keyword **IMPLICIT_DSP** *no.*

USE: Mandatory ((X.25 networks) **WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only).

TRANSPORT_PROTOCOL_ID Specifies how the NDS has to obtain the transport protocol identifier necessary in some cases to establish a transport connection in a X.25 network. (The NSAP address mapping option for X.25 networks (**WANSBKA-** profile only.) The transport protocol identifier may be entered explicitly in hexadecimal notation (**x'** *hex values***'**) or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the identifier from the DSP part *no.* of the selected remote NSAP address.

VALUE(s): Hexadecimal string (up to 16 octets).

USE: Optional

E163_NUMBER E163_INTERNATIONAL_PREFIX E163_COUNTRY_CODE E163_NATIONAL_PREFIX E163_AREA_CODE Specifies how the NDS has to obtain the complete telephone number in public and/or private networks according to CCITT recommendation E.163 (numbering plan for the international telephone service). (The NSAP address mapping options for PSTN networks (public switched telephone networks) (**WANSBKA-** profile only). The value of the **E163_NUMBER** option may be an explicit E.163 number (the entire number must be specified, including the national or international prefix, if required) or one of the keywords **IMPLICIT_IDI** or **IMPLICIT_DSP** *no.1*[**:***no.2*] may be used to cause the NDS to extract the E.163 number implicitly from the IDI (initial domain identifier) or from the DSP part *no.1* (or from the concatenation of the DSP parts *no.1* and *no.2*) of the selected remote NSAP address.

VALUE(s): 1 – 14 digits or one of the keywords **IMPLICIT_IDI** or **IMPLICIT_DSP** *no.1*[**:** *no.2*].

USE: Mandatory ((PSTN networks) **WANSBKA-** profile only).

The value of the **E163_INTERNATIONAL_PREFIX** option depends from the numbering plan used. The NDS adds the value as a prefix to the E.163 number every time an international call is made. The NDS ignores the international prefix facility (this may often be desirable in private telephone networks or if an integrated numbering is used (see also CCITT recommendation E.160) , if a dash (‘‘-’’) is assigned as a prefix value or if an E.163 number is specified explicitly by the **E163_NUMBER** option.

VALUE(s): 2 digits or dash ‘‘-’’

USE: Mandatory (only if the E.163 number is not specified explicitly).

The value of the **E163_COUNTRY_CODE** option may be the 1-3 digit CC (country code) of the country (such as 49 for Germany) in which the end system is attached to the telephone network (the list of all valid CCs is contained in annex A of recommendation E.163). The NDS removes the CC value from the called E.163 number if a telephone connection has to be established within the same country. The NDS ignores the CC facility (this may often be desirable in private telephone networks), if a dash (‘‘-’’) is assigned as a CC value or if an E.163 number is specified explicitly by the **E163_NUMBER** option.

VALUE(s): 1 – 3 digits or dash ‘‘-’’ Note: The CCs defined in annex A of recommendation E.163 do not occupy all values from the reserved range. Because the NDS compiler checks for this range only, the administrator has to guarantee the validity of the CCs entered.

USE: Mandatory (only if the E.163 number is not specified explicitly).

The value of the **E163_NATIONAL_PREFIX** option may be one or two digits (such as 0 in Germany or 0 and 1 in USA). The NDS adds this value as a prefix to the called E.163 number (by removing the CC value first) if a telephone connection has to be established within the same country but outside of the local numbering area. The NDS ignores the national prefix facility (this may often be desirable in private telephone networks), if a dash ‘‘-’’ is assigned as a prefix value or if a E.163 number is specified explicitly by the **E163_NUMBER** option.

VALUE(s): 1 – 2 digits or dash ‘‘-’’

USE: Mandatory (only if the E.163 number is not specified explicitly).

The value of the **E163_AREA_CODE** option may be the area code of the area in which the end system is attached to the telephone network (such as 89 in Munich). The NDS removes this value (by removing also the CC value) from the called E.163

number if a telephone connection has to be established within the same area. The NDS ignores the area code facility (this may often be desirable in private telephone networks), if a dash (''-'') is assigned as an area code value or if an E.163 number is specified explicitly by the **E163_NUMBER** option.

VALUE(s): 2 – 4 digits or dash ''-''

USE: Mandatory (only if the E.163 number is not specified explicitly).

The PSTN network options international prefix, country code, national prefix and area code may occur in the common options area of the XTI/CMX interface or in the mapping option area of NSAP addresses (within PSTN network relevant profiles only), whereas the E.163 address option may occur in NSAP address mapping option areas only.

E164_INTERNATIONAL_PREFIX E164_COUNTRY_CODE E164_NATIONAL_DESTINATION_CODE E164_NUMBER Specifies how the NDS has to obtain the complete ISDN number in public and/or private networks according to CCITT recommendation E.164 (numbering plan for the ISDN era). (The NSAP address mapping options for ISDN networks (integrated services digital networks) (**WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only). The value of the **E164_NUMBER** option may be an explicit E.164 number (the entire number must be specified, including the national destination code, the country code and the international prefix, if required) or one of the keywords **IMPLICIT_IDI** or **IMPLICIT_DSP** *no.1*[**:** *no.2*] may be used to cause the NDS to extract the E.164 number implicitly from the IDI (initial domain identifier) or from the DSP part *no.* (or from the concatenation of the DSP parts *no.1* and *no.2* of the selected remote NSAP address.

VALUE(s): 1 – 17 digits or one of the keywords **IMPLICIT_IDI** or **IMPLICIT_DSP** *no.1*[**:** *no.2*].

USE: Mandatory ((ISDN networks) **WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only).

The value of the **E164_INTERNATIONAL_PREFIX** option depends from the numbering plan used. The NDS adds the value as a prefix to the E.164 number everytime an international call is made. The NDS ignores the international prefix facility (this may often be desirable in private ISDN networks or if an integrated numbering is used (see also CCITT recommendation E.160)), if a dash ''-'' is assigned as a prefix value or if an E.164 number is specified explicitly by the **E164_NUMBER** option.

VALUE(s): 2 digits or dash (‘‘-’’)

USE: Mandatory (only if the E.164 number is not specified explicitly).

The value of the **E164_COUNTRY_CODE** option may be the 1-3 digit CC (country code) of the country (such as 33 for France) in which the end system is attached to the ISDN network (the list of all valid CCs is contained in annex A of recommendation E.163). The NDS removes the CC value from the called E.164 number if a ISDN connection has to be established within the same country. The NDS ignores the CC facility (this may often be desirable in private ISDN networks), if a dash ‘‘-’’ is assigned as a CC value or if an E.164 number is specified explicitly by the **E164_NUMBER** option.

VALUE(s): 1 – 3 digits or dash ‘‘-’’ Note: The CCs defined in annex A of the recommendation E.163 do not occupy all values from the reserved range. Because the NDS compiler checks for this range only, the administrator has to guarantee the validity of the CCs entered.

USE: Mandatory (only if the E.164 number is not specified explicitly).

The value of the **E164_NATIONAL_DESTINATION_CODE** option specifies the NDC part of the national (significant) number (N(S)N) which may be necessary to select a destination network. The NDS removes this value from the called E.164 number (by removing the CC value first) if an ISDN connection has to be established within the same ISDN network. The NDS ignores the national destination code facility (this may often be desirable in private ISDN networks), if a dash ‘‘-’’ is assigned as a prefix value or if a E.164 number is specified explicitly by the **E164_NUMBER** option.

VALUE(s): 1 – 15 digits or dash ‘‘-’’

USE: Mandatory (only if the E.164 number is not specified explicitly).

Notes: The E.164 options international prefix and NDC may occur in the common options area of the XTI/CMX interface or in the mapping options area of NSAP addresses (within ISDN network relevant profiles only), whereas the E.164 address option may occur in NSAP address mapping option areas only.

The E.164 options may also be used in conjunction with the X.121 options in the case of an X.31 (X.25 via ISDN) communication profile. At this, the specification of the E.164 options must be always done in front of the X.121 options. The definition of a specific E.164 international prefix- and country code option was done (although

the values are equal to the corresponding E.163 options) to make ISDN numbering independent from other numbering plans (this may be important for further extensions).

X21_DIAL_NUMBER Specifies how the NDS has to obtain the dialing number necessary for the establishment of an X.21 connection. (The NSAP address mapping option for dialup and dedicated leased lines (**WANSBKA-**, **WANNEA-** profiles only). The dialing number may be entered explicitly or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract this number from the DSP part *no.* of the selected remote NSAP address.

VALUE(s): 1 – 17 digits or the keyword **IMPLICIT_DSP** *no.*.

USE: Mandatory ((dialup) **WANSBKA-**, **WANNEA-** profiles only).

Note: This option may also be used in conjunction with the X.121 options in the case of an X.32 (X.25 via X.21) communication profile. At this, the specification of the X.21 option must be always in front of the X.121 options.

ETHERNET_ADDRESS Specifies how the NDS has to obtain the ethernet address (MAC address) required in LANs where this address information cannot be determined by means of a routing protocol (like the ARP (address resolution protocol) in TCP/IP). (The NSAP address mapping option for LAN networks (**LANSBKA-**, **OSITYPE-** profiles only). The ethernet address may be entered explicitly in hexadecimal notation (**x'***hex_values***'**) or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the address from the DSP part *no.* of the selected remote NSAP address.

VALUE(s): Hexadecimal string (always 6 octets) or keyword **IMPLICIT_DSP** *no.*.

USE Mandatory (**LANSBKA-**, **OSITYPE-** profiles only)

SUBNET_ID Specifies how the NDS has to obtain the subnet identifier required in some multiple LAN network environments. (The NSAP address mapping option for LAN networks (**LANSBKA-** profile only).The identifier may be entered explicitly or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract this identifier from the DSP part *no.* of the selected remote NSAP address. If no subnet identifier option is specified, the value 1 is assumed by the NDS.

VALUE(s): 1 – 4 digits or the keyword **IMPLICIT_DSP** *no.*.

USE: Optional (**LANSBKA-** profile only).

REGION_NUMBER PROCESSOR_NUMBER Specifies how the NDS has to obtain the region number resp. processor number required in a TRANSDATA network. (The

NSAP address mapping options for TRANSDATA networks (**WANNEA-** profile only). The region number (processor number) may be entered explicitly or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the region number (processor number) from the DSP part *no.* of the selected remote NSAP address. Note: Depending on the real communication profile (**WAN-NEA**, **WAN-NX25**, **ISDN-NEA**, **ISDN-NX25**) this set of keywords must be always used in conjunction with any of the X.121 , X.21 , E.164 or PVC options already described.

VALUE(s): 1 – 255

USE: Mandatory (**WANNEA-** profile only).

NETBIOS_HOST_NAME NETBIOS_UNIQUE_GROUP_ID Specifies how the NDS has to obtain the host name and the unique/group identifier required for a **NETBIOS** transport provider. (The NSAP address mapping options for **NETBIOS** (TCP/IP over **RFC1001**/**1002**) profile only.

The value of the host name may be entered as an ASCII string or in hexadecimal notation (**x'***hex_values* **'**)or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the host name from the DSP part *no.* of the selected remote NSAP address. The real host name provided to **NETBIOS** is always a concatenation of the host name option value and the T-selector value (called service name in this case) of the remote TSAP address.

VALUE(s): ASCII string (up to 8 characters) (excluding semicolon ‘‘;’’ and space ‘‘ ’’)or hexadecimal string (up to 8 octets). The real host name (see description above) must not exceed 16 characters (octets).

USE: Mandatory (**NETBIOS-** profile only).

As the value for the unique/group identifier one of the keywords **UNIQUE** or **GROUP** may be entered or the keyword **IMPLICIT_DSP** *no.* may be used to cause the NDS to extract the identifier from the DSP part *no.* of the selected remote NSAP address.

VALUE(s): **UNIQUE** or **GROUP**

USE: Mandatory (**NETBIOS-** profile only).

TRANSPORT_PROTOCOL_CLASS Specifies the preferred/alternative transport class values to be applied during the establishment of a transport connection. (The NSAP address mapping option for OSI transport protocol class negotiation (**WANSBKA-** profile only).

VALUE(s): 0/0 or 0/- or 2/0 or 2/2 (the first value represents the preferred class, the second value the alternative class).

USE: optional (**WANSBKA-** profile only).

LAN_CC Specifies one (or several) CC(s) which should be selected during connection establishment from the entirety of equivalent CCs to do a static load balancing or to select one of several alternative routes leading to the requested destination. (The NSAP address mapping option for routing information (CC (communication controller) selection; **LANSBKA-**, **OSITYPE-** profiles only).

VALUE(s): *CC number*[,*CC number*,..,*CC number*] (CC number = 1 – 6)

USE: optional (**LANSBKA-**, **OSITYPE-** profiles only).

WAN_CC Specifies one (or several) CC(s) and optionally one (or several) lines on a CC which should be selected during connection establishment from the entirety of equivalent CC(s) and optionally lines to do a static load balancing or to select one of several alternative routes leading to the requested destination. (The NSAP address mapping option for routing information (CC (communication controller) selection; **WANSBKA-**, **WAN3-**, **SBKA-**, **WANNEA-** profiles only).

VALUE(s): *CC number*[:*linenumber*, ..,*linenumber*] (CC number = 1 – 6; linenumber = 0 – 4 or 32 – 34; no. of option occurrences = 1 – 7)

USE: optional (**WANSBKA-**, **WAN3SBKA-**, **WANNEA-** profiles only).

H.4 NDS Compiler

The NDS compiler (**ndscomp**) compiles a NDS configuration source file and produces a NDS configuration database. The syntax of the **ndscomp** command is shown below.

ndscomp [*options*] *NDS configuration source file*

The following options are interpreted by the compiler:

-v verbose mode (causes the compiler to print debugging information and statistics to **stdout** during compilation).

-h*heap_size* Specifies, in kilobytes, the amount of dynamic allocated memory (work space) used during compilation by the compiler to temporarily store

database information. By default, if this option is not specified, the amount of work space allocated is 10 kB. Use this option whenever the default work space is insufficient to compile a configuration source file (the error message `insufficient work space memory` is printed by the compiler when this happens).

-n *net_addr* Specifies the maximum number of NSAP address descriptions occurring in the compiled configuration source file. If this option is not specified, the default value 512 is assumed by the compiler. Use this option whenever the number of NSAP address descriptions available in the configuration source file exceeds the default value (the error message `max. no. of net addresses exceeded` is printed by the compiler when this happens).

-o *DB_file* Specifies the name of the configuration database file into which the compiler may write the compiled database informations. By default, the file **NDSCONF.DIB** is created as the database file.

For compilation, any file name may be specified, but you should keep in mind that the NDS is only able to deal with a database file of name **NDSCONF.DIB**.

Notes:

- Recompile the configuration source file if any change was made. The newly created configuration database is made available to the NDS by copying the database file **NDSCONF.DIB** into the directory **/opt/dcelocal/var/adm/directory/gds/conf**. A modified configuration database becomes valid automatically for the NDS (if copied to the **conf** directory) after a short delay (1 minute). There is no need for a restart of any running application actually accessing the NDS.
- If the NDS compiler detects any problem during compilation an error message is written to **stderr**. Error messages being connected with the content of the configuration source file contain among other things the source line number at which the error was detected. Thus, by having only one statement per line, problems should be isolated very easy.

H.5 Default NDS configuration source file (NDSCONF.dat)

```
CONFIGURATION_FILE_TITLE = "SNI-configuration 25/6/93" ;
# [M] = Mandatory statement (description)
# [O] = Optional statement (description)
BEGIN SECTION INTERFACES
BEGIN INTERFACE socket
BEGIN PROFILE RFC1006   # TCP/IP over RFC1006
PROFILE_PRIORITY = 1 ;
# OSI-NSAP addresses with IP-address, port-no. and T-selector
# information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [M]   INTERNET_ADDRESS = IMPLICIT_DSP <dsp-part no.> |
#                          <IP-address (dotted notation)> ;
# [M/O] PORT = IMPLICIT_DSP <dsp-part no.> | <port no.> ;
# [M/O] T_SELECTOR = <T-selector (ASCII)> | x'<T-selector (hex)>' ;
#
# additional NSAP-addr. and mapping options must be inserted here
#
# -- ---- ---- ---- ---- ---- ----
# address format 1 (RFC1277)
# -- ---- ---- ---- ---- ---- ----
#       IDP    |                 DSP
# +---+-- ---- --+-- ----+-- ---- ----+-- --+-- ---- ---- ---+
# |AFI|  IDI   |prefix|IP-address|port|transport set|
# | 54|00728722|  03  |          |    |   00001     |
# +---+-- ---- --+-- ----+-- ---- ----+-- --+-- ---- ---- ---+
#
NSAP_ADDRESS_GROUP = 540072872203[0-9]{12}[0-9]{5}(00001{0,1}) ;
INTERNET_ADDRESS = IMPLICIT_DSP 2 ;
PORT = IMPLICIT_DSP 3 ;
            # -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 2
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP   |                       DSP
```

```
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- --+
# |AFI|IDI |DFI|SFI|country|res.|location|area|IP-address|NSEL|
# | 47|0058| 01| 05|       |0000|        |    |          |    |
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- --+
#
NSAP_ADDRESS_GROUP =
4700580105[0-9A-F]{4}0000[0-9A-F]{4}[0-9A-F]{4}[0-9]{12}[0-9A-F]{2} ;
INTERNET_ADDRESS = IMPLICIT_DSP 7 ;
PORT = 21010 ; T_SELECTOR = GDS-CLIENT ;
PORT = 21011 ; T_SELECTOR = GDS-SERVER1 ;
PORT = 21012 ; T_SELECTOR = GDS-SERVER2 ;
PORT = 21013 ; T_SELECTOR = GDS-SERVER3 ;
END PROFILE RFC1006
END INTERFACE socket
BEGIN INTERFACE xti
# common profile options
PROFILE_DEVICE = t_msg ;
# [M/O] X121_INTERNATIONAL_PREFIX = <international prefix> ;
X121_INTERNATIONAL_PREFIX = 0 ;
# [M/O] X121_DATA_COUNTRY_CODE = <data country code> ;
#       or
#       X121_DATA_NETWORK_ID_CODE = <data network-ID code> ;
X121_DATA_COUNTRY_CODE = 262 ;
# [M/O] E163_INTERNATIONAL_PREFIX = <international prefix> ;
E163_INTERNATIONAL_PREFIX = 00 ;
# [M/O] E163_COUNTRY_CODE = <country code> ;
E163_COUNTRY_CODE = 49 ;
# [M/O] E163_NATIONAL_PREFIX = <national prefix> ;
E163_NATIONAL_PREFIX = 0 ;
# [M/O] E163_AREA_CODE = <area code> ;
E163_AREA_CODE = 89 ;
# [M/O] E164_INTERNATIONAL_PREFIX = <international prefix> ;
E164_INTERNATIONAL_PREFIX = 00 ;
# [M/O] E164_COUNTRY_CODE = <country code> ;
E164_COUNTRY_CODE = 49 ;
# [M/O] E164_NATIONAL_DESTINATION_CODE = <national destination code> ;
E164_NATIONAL_DESTINATION_CODE = 9999 ;
#       BEGIN PROFILE LOOPSBKA  # local loopback
#           PROFILE_PRIORITY = 9 ;
# OSI-NSAP addresses without additional mapping options
```

```
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
#
# additional NSAP-addresses must be inserted here
#
#       END PROFILE LOOPSBKA
BEGIN PROFILE OSITYPE   # ETHN-CLNS-active
PROFILE_PRIORITY = 8 ;
# OSI-NSAP addresses with MAC-address and CC-routing information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [O]   ETHERNET_ADDRESS = IMPLICIT_DSP <dsp-part no.> |
#                          x'<MAC-address (hex)>' ;
# [O]   LAN_CC = <ccnumber>[,<ccnumber>, ... ,<ccnumber>] ;
            #
# additional NSAP-addr. and mapping options must be inserted here
#
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 1
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP   |                        DSP
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
# |AFI|IDI |DFI|SFI|country|res.|location|area|MAC-address|NSEL|
# | 47|0058| 01| 01|       |0000|        |    |           |    |
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
#
NSAP_ADDRESS_GROUP = 4700580101[0-9A-F]{4}0000[0-9A-F]{22} ;
ETHERNET_ADDRESS = IMPLICIT_DSP 7 ;
END PROFILE OSITYPE
BEGIN PROFILE LANSBKA   # ETHN-CLNS-inactive/TR-LLC
PROFILE_PRIORITY = 7 ;
# OSI-NSAP addresses with MAC-address and CC-routing information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [M]   ETHERNET_ADDRESS = IMPLICIT_DSP <dsp-part no.> |
#                          x'<MAC-address (hex)>' ;
# [O]   SUBNET_ID = IMPLICIT_DSP <dsp-part no.> |
```

```
#                   <subnet-ID> ;
# [O]   LAN_CC = <ccnumber>[,<ccnumber>, ... ,<ccnumber>] ;
#
# additional NSAP-addr. and mapping options must be inserted here
#
            # -- ---- ---- ---- ---- ---- ---- ---- ----
# address format 1 (historic format)
# -- ---- ---- ---- ---- ---- ---- ---- ----
#  IDP|          DSP
# +---+-- ---- ---+-- ---- -----+-- --+
# |AFI|subnet-ID|MAC-address|NSEL|
# | 49|         |           |    |
# +---+-- ---- ---+-- ---- -----+-- --+
#
NSAP_ADDRESS_GROUP = 49[0-9A-F]{4}[0-9A-F]{12}[0-9A-F]{2} ;
ETHERNET_ADDRESS = IMPLICIT_DSP 2 ;
SUBNET_ID = IMPLICIT_DSP 1 ;
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 2
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP   |                       DSP
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
# |AFI|IDI |DFI|SFI|country|res.|location|area|MAC-address|NSEL|
# | 47|0058| 01| 01|       |0000|        |    |           |    |
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
#
#           NSAP_ADDRESS_GROUP = 4700580101[0-9A-F]{4}0000[0-9A-F]{22} ;
#           ETHERNET_ADDRESS = IMPLICIT_DSP 7 ;
END PROFILE LANSBKA
BEGIN PROFILE WANSBKA   # WAN-CONS, ISDN-CONS
PROFILE_PRIORITY = 5 ;
# OSI-NSAP addresses with X.121-, E.163-, E.164-, X.21-, PVC-number,
# transport protocol-ID, transport class and CC-routing information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [M]   X121_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <X.121-number> ;
# [M/O] X121_INTERNATIONAL_PREFIX = <international prefix> ;
```

```
# [M/O] X121_DATA_COUNTRY_CODE = <data country code> ;
#       or
# [M/O] X121_DATA_NETWORK_ID_CODE = <data network-ID code> ;
#
# or
#
# [M]   E163_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <E.163-number> ;
# [M/O] E163_INTERNATIONAL_PREFIX = <international prefix> ;
# [M/O] E163_COUNTRY_CODE = <country code> ;
# [M/O] E163_NATIONAL_PREFIX = <national prefix> ;
# [M/O] E163_AREA_CODE = <area code> ;
#
# or
#
# [M]   E164_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <E164-number> ;
# [M/O] E164_INTERNATIONAL_PREFIX = <international prefix> ;
# [M/O] E164_COUNTRY_CODE = <country code> ;
# [M/O] E164_NATIONAL_DESTINATION_CODE = <national dest. code> ;
#
# or
#
# [M]   PVC_CHANNEL = IMPLICIT_DSP <dsp-part no.> |
#                     <pvc-number> ;
#
# or
#
# [M]   X21_DIAL_NUMBER = IMPLICIT_DSP <dsp-part no.> |
#                         <X.21-number> ;
#
# and
#
# [O]   WAN_CC = <ccnumber>[:<linenumber>, ... ,<linenumber>] ;
# [O]   TRANSPORT_PROTOCOL_ID = x'<transport protocol-ID (hex)>' ;
# [O]   TRANSPORT_PROTOCOL_CLASS = <pref. class>/<alt. class> ;
#
# additional NSAP-addr. and mapping options must be inserted here
```

```
#
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 1: AFI = 36,37,52,53
# -- ---- ---- ---- ---- ---- ---- ---- -----
#        IDP
# +---+-- ---- ---- --+
# |AFI|    IDI     |
# | xx|X.121-number|
# +---+-- ---- ---- --+
#
NSAP_ADDRESS_GROUP = 3[67][0-9]{1,14} ;
X121_NUMBER = IMPLICIT_IDI ;
NSAP_ADDRESS_GROUP = 5[23][0-9]{1,14} ;
X121_NUMBER = IMPLICIT_IDI ;
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 2: AFI = 42,43,56,57
# -- ---- ---- ---- ---- ---- ---- ---- -----
#        IDP
# +---+-- ---- ---- --+
# |AFI|    IDI     |
# | xx|E.163-number|
# +---+-- ---- ---- --+
#
NSAP_ADDRESS_GROUP = 4[23][0-9]{1,12} ;
E163_NUMBER = IMPLICIT_IDI ;
NSAP_ADDRESS_GROUP = 5[67][0-9]{1,12} ;
E163_NUMBER = IMPLICIT_IDI ;
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 3: AFI = 44,45,58,59
# -- ---- ---- ---- ---- ---- ---- ---- -----
#        IDP
# +---+-- ---- ---- --+
# |AFI|    IDI     |
# | xx|E.164-number|
# +---+-- ---- ---- --+
#
NSAP_ADDRESS_GROUP = 4[45][0-9]{1,15}F ;
E164_NUMBER = IMPLICIT_IDI ;
NSAP_ADDRESS_GROUP = 5[89][0-9]{1,15}F ;
E164_NUMBER = IMPLICIT_IDI ;
```

```
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 4
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP   |                        DSP
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
# |AFI|IDI |DFI|SFI|country|res.|location|DCC/|DTE-address|NSEL|
# | 47|0058| 01| 06|       |0000|        |DNIC|           |    |
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
#
NSAP_ADDRESS_GROUP = 4700580106[0-9A-F]{4}0000[0-9A-F]{22} ;
X121_NUMBER = IMPLICIT_DSP 6:7 ;
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 5
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP   |                        DSP
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
# |AFI|IDI |DFI|SFI|country|res.|location| CC |NSN(NDC/SN)|NSEL|
# | 47|0058| 01| 07|       |0000|        |    |           |    |
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- --+-- ---- ----+-- -+
#
NSAP_ADDRESS_GROUP = 4700580107[0-9A-F]{4}0000[0-9A-F]{22} ;
E164_NUMBER = IMPLICIT_DSP 6:7 ;
# -- ---- ---- ---- ---- ---- ----
# address format 6 (RFC1277)
# -- ---- ---- ---- ---- ---- ----
#       IDP    |              DSP
# +---+-- ---- --+-- ----+-- --+-- ----+---+-- ---- -----+
# |AFI|  IDI   |prefix|type|length|PID|DTE-address|
# | 54|00728722|  01  |  1 |      |   |           |
# +---+-- ---- --+-- ----+-- --+-- ----+---+-- ---- -----+
#
NSAP_ADDRESS_GROUP = 5400728722011[0-9]{5,27} ;
X121_NUMBER = IMPLICIT_DSP 4 ;
TRANSPORT_PROTOCOL_ID = IMPLICIT_DSP 3 ;
            # -- ---- ---- ---- ---- ---- ---- ---- ----
# address format 7 (historic format)
# -- ---- ---- ---- ---- ---- ---- ---- ----
#       IDP    |              DSP
# +---+-- ---- --+-- ----+-- ---- -----+---+
# |AFI|  IDI   |prefix|DTE-address|PID|
```

```
# | 54|00521090|  01  |           |   |
# +---+-- ---- --+-- ----+-- ---- -----+---+
#
NSAP_ADDRESS_GROUP = 540052109001[0-9]{14}[0-9]{12} ;
X121_NUMBER = IMPLICIT_DSP 2 ;
TRANSPORT_PROTOCOL_ID = IMPLICIT_DSP 3 ;
END PROFILE WANSBKA
#       BEGIN PROFILE WAN3SBKA  # WAN-X25, ISDN-X25, ISDN-HDLC
#           PROFILE_PRIORITY = 3 ;
# OSI-NSAP addresses with X.121-, E.164-, PVC-number and
# CC-routing information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [M]   X121_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <X.121-number> ;
# [M/O] X121_INTERNATIONAL_PREFIX = <international prefix> ;
# [M/O] X121_DATA_COUNTRY_CODE = <data country code> ;
#       or
# [M/O] X121_DATA_NETWORK_ID_CODE = <data network-ID code> ;
#
# or
#
# [M]   E164_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <E164-number> ;
# [M/O] E164_INTERNATIONAL_PREFIX = <international prefix> ;
# [M/O] E164_COUNTRY_CODE = <country code> ;
# [M/O] E164_NATIONAL_DESTINATION_CODE = <national dest. code> ;
#
# or
#
# [M]   PVC_CHANNEL = IMPLICIT_DSP <dsp-part no.> |
#                     <pvc-number> ;
#
# and
#
# [O]   WAN_CC = <ccnumber>[:<linenumber>, ... ,<linenumber>] ;
#
```

```
# additional NSAP-addr. and mapping options must be inserted here
#
# see profile WANSBKA
#       END PROFILE WAN3SBKA
BEGIN PROFILE WANNEA    # WAN-NEA, WAN-NX25, ISDN-NEA, ISDN-NX25
PROFILE_PRIORITY = 4 ;
# OSI-NSAP addresses with region/processor-,X.121-, E.164-,
# X.21-, PVC-number and CC-routing information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [M]   PROCESSOR_NUMBER = IMPLICIT_DSP <dsp-port no.> |
#                          <processor-number> ;
# [M]   REGION_NUMBER = IMPLICIT_DSP <dsp-port no.> |
#                       <region-number> ;
#
# and
#
# [M]   X121_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <X.121-number> ;
# [M/O] X121_INTERNATIONAL_PREFIX = <international prefix> ;
# [M/O] X121_DATA_COUNTRY_CODE = <data country code> ;
#       or
# [M/O] X121_DATA_NETWORK_ID_CODE = <data network-ID code> ;
#
# or
#
# [M]   E164_NUMBER = IMPLICIT_IDI |
#                     IMPLICIT_DSP <dsp-part no.> |
#                     <E164-number> ;
# [M/O] E164_INTERNATIONAL_PREFIX = <international prefix> ;
# [M/O] E164_COUNTRY_CODE = <country code> ;
# [M/O] E164_NATIONAL_DESTINATION_CODE = <national dest. code> ;
#
# or
#
# [M]   X21_DIAL_NUMBER = IMPLICIT_DSP <dsp-part no.> |
#                         <X.21-number> ;
#
```

```
# or
#
# [M]   PVC_CHANNEL = IMPLICIT_DSP <dsp-part no.> |
#                     <pvc-number> ;
#
# and
#
# [O]   WAN_CC = <ccnumber>[:<linenumber>, ... ,<linenumber>] ;
#
# additional NSAP-addr. and mapping options must be inserted here
#
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 1
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP   |                        DSP
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- -----+-- ---- ---+-- -+
# |AFI|IDI |DFI|SFI|country|res.|NEA-net-|region-|processor-|NSEL|
# | 47|0058| 01| 01|       |0000| number | number|  number  |    |
# +---+-- --+---+---+-- -----+-- --+-- ---- --+-- -----+-- ---- ---+-- -+
            #
NSAP_ADDRESS_GROUP =
4700580102[0-9A-F]{4}0000[0-9A-F]{4}00[0-9A-F]{2}0{10}[0-9A-F]{2}00 ;
PROCESSOR_NUMBER = IMPLICIT_DSP 7 ;
REGION_NUMBER = IMPLICIT_DSP 6 ;
X121_NUMBER = 99999999 ;    # unreal statement/value (should
# be replaced by a real one)
END PROFILE WANNEA
BEGIN PROFILE NETBIOS
PROFILE_PRIORITY = 6 ;
PROFILE_DEVICE = /dev/inet/nb ;
# OSI-NSAP addresses with NETBIOS-host name and NETBIOS-group/
# unique information
# [M]   NSAP_ADDRESS = <NSAP-address (macro format)> ;
#       or
#       NSAP_ADDRESS_GROUP = <NSAP-addresses (regular expression)> ;
# [M]   NETBIOS_HOST_NAME = IMPLICIT_DSP <dsp-part no.> |
#                           <NETBIOS-hostname (ASCII)> ;
# [M]   NETBIOS_UNIQUE_GROUP_ID = IMPLICIT_DSP <dsp-part no.> |
#                                 UNIQUE | GROUP ;
#
```

```
# additional NSAP-addr. and mapping options must be inserted here
#
# -- ---- ---- ---- ---- ---- ---- ---- -----
# address format 1
# -- ---- ---- ---- ---- ---- ---- ---- -----
#    IDP       |                DSP
# +---+-- ---- --+--+-- ---- ---+-- ---- ---- ---- --+-- ---- --+
# |AFI|  IDI   |FI|subnet-ID|NETBIOS-hostname|unique/ |
# | 41|50093994|01|         |                |group-ID|
# +---+-- ---- --+--+-- ---- ---+-- ---- ---- ---- --+-- ---- --+
            #
NSAP_ADDRESS_GROUP = 415009399401[0-9A-F]{4}[0-9A-F]{16}0[01] ;
NETBIOS_HOST_NAME = IMPLICIT_DSP 3 ;
NETBIOS_UNIQUE_GROUP_ID = IMPLICIT_DSP 4 ;
END PROFILE NETBIOS
END INTERFACE xti
END SECTION INTERFACES
```

Appendix I

Navigation in the GDS

This appendix describes information about the internal behavior of GDS. The material is provided for more advanced administrators who want to know detailed information on how Directory Service requests are routed and processed among DSAs.

I.1 Continuation References

A continuation reference describes how a service request can be continued at one or more other DSAs. A continuation reference is typically returned when a DSA is unable to fulfill the request itself.

A continuation reference has the following structure:

TARGET OBJECT
: The DN of the target object of the continued request or subrequest. In general the target object of the continued request is the same as the target object of the incoming request.

The **TARGET OBJECT** is not the DN of the target object of the incoming request if the DSA had dereferenced an alias. For example, suppose that the **TARGET OBJECT** of the incoming request is:

$/RDN_1/.../RDN_k/RDN_{k+1}/.../RDN_n$

and that **/RDN**$_k$ is an alias name for the object:

$/RDN_1,/.../RDN_j,$

then the target object would be:

$/RDN_1,/.../RDN_j,/RDN_{k+1}/.../RDN_n$

The **TARGET OBJECT** is not the DN of a subordinate of the incoming request if the following conditions are met:

- It is a **search** or **list** operation
- The DSA has administrative authority for the target object
- There is at least one immediate subordinate of it which is mastered by another DSA

For example, suppose the target object of the incoming request is:

$/RDN_1/.../RDN_m$

The target object of the continuation reference is:

$/RDN_1/.../RDN_m/.../RDN_n$

The DSA is master of this object and is not master of the subordinate object:

$/RDN_1/.../RDN_m/.../RDN_n/RDN_{n+1}$

ALIASED RDNS
The number of RDNs of the target object that have been generated by dereferencing an alias. **ALIASED RDNS** is present only if an alias was dereferenced. In the example above, the **ALIASED RDNS** would be simply *j*.

OPERATION PROGRESS
Describes the progress of the name resolution as one of the following:

NOT STARTED
The DSA has no knowledge of the target object nor any of its superior nodes.

PROCEEDING
The DSA has knowledge about the target object of any of its superior nodes (indicated by the **next RDN to be resolved**). The DSA that receives the request should have administrative authority of this object, but not of its superior node.

COMPLETED
Name resolution is completed. The DSA that gets the continued request is no longer involved in finding the target object.

REFERENCE TYPE
Characterizes the type of the reference as one of the following:

SUPERIOR Set when the DSA has no knowledge of the target object or any of its superior nodes

SUBORDINATE
Operation progress is proceeding;(let *k* be the next RDN to be resolved),the DSA has administrative authority of **RDN** $_{k-1}$.

CROSS Operation progress is proceeding;(let *k* be the next RDN to be resolved),the DSA does not have administrative authority of $\mathbf{RDN}_{k-1}$.

NONSPECIFIC SUBORDINATES
Not used in GDS

RDNS RESOLVED
: Specifies the last RDN of the target object that is mastered by the DSA; it is present when **REFERENCE TYPE** is **CROSS**.

ENTRY ONLY
: Boolean component which is **TRUE** if, in a one-level search, the DSA has found an alias as a subordinate of the base object whose aliased object could not be found locally.

ACCESS POINTS
: A set of DN and PSAP address pairs of the DSA that should continue to perform the operation. The continuation references that are generated by the GDS contain only one such access point.

The continuation references are returned to the DUA either as a referral or as components of a partial outcome qualifier in the result of a **list** or **search** operation. The DUA may use the access points, to bind the referenced DSAs and to forward the request to them. The DUA includes the **OPERATION PROGRESS** and the **ALIASED RDNS** components in the common arguments to inform the referenced DSA about the status of the operation.

When the DSA handles the continuation references, it includes all the components except **RDNS RESOLVED** and **ACCESS POINTS** into the chaining arguments of the chained request or subrequest to inform the referenced DSA about the status of the operation. The access points are used to bind the referenced DSAs by DSP.

I.2 Generating References from the Local Database

This section describes how references are generated from information from the local database.

Every entry of the local database contains the **MASTER KNOWLEDGE** attribute. The **MASTER KNOWLEDGE** attribute contains the DN of the DSA that has the administrative authority of the entry. If this DN is different from the name of the performing DSA, it may be used for a continuation reference. To generate the ACCESS POINT, the DSA reads the presentation address of this DSA from its local database. The DSA must contain this entry, otherwise it cannot work cooperatively with this DSA.

References are generated in the following phases:

Name Resolution
: The handling of the request, before the target object of the operation has been found.

Evaluation
: Performs the operation specified by a particular directory request (for example, a **search** request) after the target object of the operation has been found.

To demonstrate how references are generated in the Name Resolution phase, suppose the target object is:

$RDN_1/.../RDN_n$

The DSA searches for the longest part of the target object in its local database and finds the entry:

$RDN_1/.../RDN_m$, $m <= n$

This entry may be an object mastered by the DSA itself, a shadow or an alias.

The following situations can occur:

m = **0**
: The DSA does not even hold $\mathbf{RDN}_1$. The DSA checks whether it is a first-level DSA. If so, the DSA generates a name error with the problem **NO SUCH OBJECT**. Otherwise the DSA generates a reference with the components shown in Table A-1.

0 < *m* < *n*
: The DSA holds $\mathbf{RDN}_1/.../RDN_m$, but not $\mathbf{RDN}_{m+1}$ (nor the target object itself). If this object is mastered by the DSA itself, it creates a name error with the problem **NO SUCH OBJECT**. If the object is mastered by another DSA (for example, **DSA1**) it determines the following and creates a reference with the components shown in Table A-2:

  - the last $\mathbf{RDN}_k$ that names an object that is mastered by a DSA, that is different from **DSA1**
  - the last $\mathbf{RDN}_j$ that is mastered by the DSA itself

Suppose the object is an alias, and **RDN1'/.../RDN$_k$'** is the aliased object name. If the operation is a modify operation, or if it is a retrieval operation, and the service control bit **DON'T DEREFERENCE ALIAS** is set, a name error with problem **ALIAS DEREFERENCING PROBLEM** is generated. Otherwise the target object will simply change to **RDN'/.../RDN$_k$'/RDN** $_{m+1}$/.../**RDN$_n$**.

The procedure of name resolution restarts with this target object. If it terminates with a reference (as described in other cases), it contains the component **ALIASED RDNS** = **k**.

m = n The DSA holds an entry of the target object. If it is a master entry, the name resolution terminates with success, no references are generated.

If the entry is a shadow, the service control option **DON'T USE COPY** is not set, and the operation is a single object operation, name resolution terminates and no references are generated.

If the entry is a shadow, the service control option **DON'T USE COPY** is not set, the operation is a **list** or **search** operation, and the service control option **LOCAL SCOPE** is set, name resolution terminates and no references are generated. Otherwise a continuation reference is generated as described in the previous case.

Table I–1. Reference Component Values When m=0

Component	**Value**
Target Object	*target object of incoming request*
Aliased RDNs	not present
Operation Progress	**not started**
RDNs Resolved	not present
Reference Type	**superior**
Access Points	derived from master know-
	ledge for first level object that is superior to its own DSA object
Entry Only	not present

Table I–2. Reference Component Values When 0<m<n

Component	Value
Target Object	*target object of incoming request*
Aliased RDNs	not present
Operation Progress	**proceeding** (the next RDN to be resolved is *k+1*)
RDNs Resolved	*j*
Reference Type	**subordinate** (only if *j =k*); or **cross**
Access Points	derived from master knowledge for $\mathbf{RDN}_1$/...RDN_{k+1}
Entry Only	not present

In the phase of request decomposition references are generated when a partial result has been generated locally, but the DSA has some knowledge of other DSAs holding information that is needed to satisfy the request completely.

A reference is generated for a subordinate of the target object of the incoming request $\mathbf{RDN}_1$/.../RDN_n **/.../**$\mathbf{RDN}_p$ (let **S** be a symbolic name for $\mathbf{RDN}_1$/.../RDN_n **/.../**$\mathbf{RDN}_p$) when:

- **S** is a shadow.
- The immediate superior node of **S** is not a shadow.
- The reference is not a replica of another reference that has been generated previously. This may happen if the DSA holds shadows that share the superior node and the master knowledge.

The references have the components shown in Table A-3.

Table I–3. Reference Component Values for a Subordinate of a Base Object

Component	Value
Target Object	$\mathbf{RDN}_1$/.../RDN_{p-1}
Aliased RDNs	not present
Operation Progress	**completed**

Component	Value
RDNs Resolved	not present
Reference Type	**subordinate**
Access Points	derived from master knowledge of **RDN**$_1$/.../RDN$_p$
Entry Only	**TRUE** if the operation is a search with subset **ONE LEVEL**, **SEARCH ALIASES** is **TRUE**, and the target object is the aliased object of an immediate subordinate of the base object; or **FALSE** otherwise

Appendix J

The DSA configuration file

The DSA configuration file is a text file which has the pathname **opt/dcelocal/var/directory/gds/dsa/dir** *id* **/dsaconf**, where *id* is the directory ID. It must be generated by the Directory system administrator manually using a text editor and it is read by the DSA processes during initialization. It contains various segments of information, that are recognized by keywords. If the configuration file is missing or has a format that cannot be understood by the DSA, default values are assumed. Errors from reading the DSA configuration file are logged into the DSA logfile. The administrator should examine the contents of the logfile of a booting DSA after generating the DSA configuration file to check for errors. The keywords and the format of the information attached to them are as follows:

CHAINING_DSA SINGLE_HOP | MULTI_HOP Describes the chaining policy of the DSA. If the DSA is configured for **SINGLE_HOP** chaining, it will not chain any chained request in general, but will return referrals in name resolution or a partial outcome qualifier with continuation references in request decomposition. Other requests are chained if necessary and specified by service controls appropriately. If the DSA is configured for **MULTI_HOP** chaining, it will chain every request if necessary and specified by service controls appropriately. The default value is **MULTI_HOP**.

ROUTING_INFO [ROUTING][KNOWLEDGE] \ [ROUTING_DSAS *n* dsa_1 [... dsa\$_n$]] Specifies whether the DSA may be used as a routing DSA, and which information should be used to get the DSAs to which chained requests should be routed. **ROUTING** specifies whether the DSA may be used as a **ROUTING DSA**. If a DSA is configured as a **ROUTING DSA**, it accepts chained requests that are designated for other DSAs and routes them to the correct DSAs. **KNOWLEDGE** specifies that the DSA chains the requests to the access points that are known from its own knowledge base. **ROUTING_DSAS** specifies that the DSA chains the requests to the access points that are specified by the DSA names of the given list. *n* specifies the number of routing DSAs in the list. The DSA names must be in the following syntax: RDN_1/.../RDN\$_n$ The RDNs must be in the following syntax: ob_id_1=val_1 ,...,ob_id\$_n$=val\$_n$ where *ob_idi* denotes the object identifier of an attribute type, *val* denotes the assigned value. The DSA names must be separated by new lines. No empty lines between the DSA names are accepted. The ordering of the two items **KNOWLEDGE** and **ROUTING_DSAS** and (if present) the ordering of the DSAs in the list define the ordering that the DSA uses for chaining. If one DSA cannot be bound, the next DSA in the list will get the chained request. The default value is **KNOWLEDGE**.

DSP_UNBIND_DELAY *minutes minutes* is the delay time in minutes by which the DSP unbind should be delayed after a result was received from an association. The default value is **0**, indicating that the unbind is issued immediately after the desired result was returned. This information is read and used by **gdsdsa**.

After creating the DSA configuration file the administrator has to reactivate the directory to make the configuration parameters known to the DSA.

The following is an example of the syntax of a **dsaconf** file:

```
CHAINING_DSA SINGLE_HOP
DSP_UNBIND_DELAY 2
ROUTING_INFO
ROUTING_DSAS 3
85.4.6=DE/85.4.10=SNI/85.4.11=BUBA NM 12/85.4.3=DSA/85.4.3=DSA5
85.4.6=DE/85.4.10=SNI/85.4.11=BUBA NM 12/85.4.3=DSA/85.4.3=DSA6
85.4.6=DE/85.4.10=SNI/85.4.11=BUBA NM 12/85.4.3=DSA/85.4.3=DSA7
```

Appendix K

The XOI_SCHEMA_FILE

This appendix describes the **XOI_SCHEMA_FILE** file.

A schema specification file is used by **gdscp** to read some of the details related to attributes and object classes in the schema. This schema file is normally read from the installation directory defined by the environment variable **dcelocal_path** in **dce.h**. However, you can specify your own schema file by setting the value of the environment variable **XOI_SCHEMA_FILE**.

If you change the schema, you must update the schema specification manually using a text editor. **gdscp** used this file when displaying information. If you do not update the file to reflect the current schema, the attributes displayed by **gdscp** will not accurately reflect the current schema attributes.

The schema specification file consists of the following main sections:

1. The Object Class Definition Block
2. The Attribute Definition Block
3. A series of OM Class Definition Blocks

Use the following format to change the Object Class Definition Block (>> indicates the start of a new line; > indicates the start of a new word):

```
>> {
>>Object class abbreviation
>Full name of the object Class
>Object identifier string of the object class
>> }
```

Use the following format to change the Attribute Definition Block:

```
>> {
>>Attribute abbreviation
>Full name of the attribute
>Object identifier string of the attribute
>Attribute syntax as defined by XOM
>OM class name as defined by XOM
>> }
```

Use the following format to change the OM Class Definition Blocks:

```
>> {
>>Component abbreviation
>Component full name
>OMK type of the component as defined by XOM
>Syntax of the component as defined by XOM
>OM class name of the component
>> }
```

The following example shows a partial schema specification file. The definition blocks have been shortened for clarity. The entire schema specification file would contain multiple pages of information.

```
#Object Class Definition Block
{
C    Country                    85.6.2
ORP  Organizational-Person      85.6.7
}
#Attribute Definition Block
{
OCL Object-Class    85.4.0  OM_S_OBJECT_IDENTIFIER_STRING  0
C   Country         85.4.6  OM_S_PRINTABLE_STRING          0
CN  Common-Name     85.4.3  OM_S_TELETEX_STRING            0
PA  Postal-Address  85.4.16 OM_S_OBJECT                    DS_C_POSTAL_ADDRESS
TXN Telex-Number    85.4.21 OM_S_OBJECT                    DS_C_TELEX_NBR
}
#OM Class Definition Block for Postal-Address
{
DS_C_POSTAL_ADDRESS
PA  Postal-Address   806         OM_S_TELETEX_STRING          0
}
#OM Class Definition Block for Telex-Number
{
DS_C_TELEX_NBR
TXN Telex-Number     809         OM_S_PRINTABLE_STRING       0
TXC Country-Code     802         OM_S_PRINTABLE_STRING       0
TXA Answerback       801         OM_S_PRINTABLE_STRING       0
}
#OM Class Definition Block for DSX_C_GDS_ACL
{
DSX_C_GDS_ACL        859   OM_S_OBJECT                       DSX_C_GDS_ACL_ITEM
PBM Modify-Public    860   OM_S_OBJECT                       DSX_C_GDS_ACL_ITEM
STR Read-Standard    861   OM_S_OBJECT                       DSX_C_GDS_ACL_ITEM
SER Read-Sensitive   862   OM_S_OBJECT                       DSX_C_GDS_ACL_ITEM
SEM Modify-Sensitive 863   OM_S_OBJECT                       DSX_C_GDS_ACL_ITEM
}
#OM Class Definition Block for DSX_C_GDS_ACL_ITEM
{
INT Interpretation   864   OM_S_ENUMERATION                  0
USR User             865   OM_S_OBJECT                       DS_C_DS_DN
}
```

Format the schema specification file by using the following rules:

- All lines starting with the number sign (#) are treated as comment lines.
- The three types of blocks must be in the order shown above.
- Use a space as a delimiter between fields.
- All abbreviations within a Definition Block must be unique.
- The value of the OM class name field for all unstructured attributes in the Attribute Definition Block must 0.
- The **Attribute Syntax** field for all structured attributes must be **OM_S_OBJECT**.
- The value for the **OM Class Name** field for unstructured components must be 0.
- The value for the **Syntax** field in the OM Class Definition Block must be **OM_S_OBJECT**.

The following rules apply to the **Object Class Definition Block**:

- The object class identifier must begin with an alphabetic character and can contain only alphanumeric characters.
- The object identifier string must be a sequence of digits separated by dots (.).
- Multiple occurrences of object identifier strings and abbreviations are not valid.

The following rules apply to the Attribute Definition Block:

- The attribute abbreviation must begin with an alphabetic character and can contain only alphanumeric characters.
- The attribute object identifier string must be a sequence of digits separated by dots (.).
- The attribute syntax must be a valid OM syntax.
- The class name, if present, must be valid.
- Multiple occurrences of object identifier strings and abbreviations are not valid.

The following rules apply to the OM Class Definition Block:

- The class name must be a valid name.
- The component name must begin with an alphabetic character and can contain only alphanumeric characters.

- The syntax must be a valid OM syntax.
- Multiple occurrences of an OM class name are not valid.
- An OM Class Definition Block for an OM class name must be present in the Attribute Definition Block or in another OM Class Definition Block defined before it in the file.

Certain checks are performed on all the field elements of the file. If any error is encountered during these checks, the reading of the file is aborted and all the XOI routines return the error **XOI_SCHEMA_NOT_READ**.

Part 2

OSF DCE GDS Administration Reference

Chapter 12

Administrative Commands

gds_intro

Purpose Introduction to commands for Global Directory Service (GDS) administration

DESCRIPTION

The following GDS administration commands are available:

gdssysadm Calls the GDS system administration.

gdsditadm Calls the Directory Information Tree (DIT) administration.

gdscacheadm
Calls the cache administration.

gdscp Calls a command-line interface for object administration.

The following configuration and initialization program is available:

gdssetup Simplifies the configuration and initialization of the Directory Service.

The following GDS log file evaluation command is available:

gdsstep Evaluates the log files of GDS processes.

The following GDS monitoring command is available:

gdsdirinfo Displays information on the status of GDS daemon processes and application processes that use GDS.

The following IPC-resource maintenance and monitoring commands are available:

gdsipcinit Provides and removes IPC resources.

gdsipcstat Displays internal IPC information.

gdscacheadm

Purpose

Cache administration program

Synopsis

gdscacheadm [**-i** *directory ID*] [**-o** *operation*] [*output*] [<*input*]

Options

-i *directory ID*
The directory ID. Valid values are 1 to 20.

-o *operation* The administration function. Valid values are

1 Object administration

2 Cache update

If this option is not specified, the **gdscacheadm** program outputs Mask 3 (see the *DCE 1.2.2 GDS Administration Guide and Reference*).

Arguments

output Name of the output file in batch mode. The output file has a simple filename (that is, it cannot contain directory pathnames). The output file will be created with the name specified in the **/opt/***dcelocal*/**var/adm/directory/gds/adm** directory. If no *output* parameter is specified in batch mode, then the file **/opt/***dcelocal*/**var/adm/directory/gds/adm/tstfile** is created.

input Name of the input file in batch mode. For the *input* argument, any UNIX filename is valid. As it is a redirection, the file name is interpreted by the shell.

Description

The **gdscacheadm** shell command is used to call the cache administration. The options **-i** and **-o** are only possible in dialog mode, and *output* and *input* are only used in batch mode.

Structure of the Input File in Batch Mode

An input file is required for the directory administration in batch mode. This file has to be provided with the mask entries in the same mask sequence as for dialog mode.

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 are overwritten correctly.

Comments can be inserted and must be enclosed between : (colons).

Comments that begin with ‘‘:*’’ (double quote, colon, asterisk, double quote) and end with ‘‘*:’’ are transferred from the input file to the output file.

Structure of the Output File in Batch Mode

The output file in batch mode contains the comments transferred from the input file and the program messages displayed in the masks. Error messages are thus indicated by the character string ERROR.

If the administration program is called with no further option, it outputs the Logon Menu Mask. (See the *DCE 1.2.2 Administration Guide*.)

This is an application of the GDS. Hence, serviceability messages are routed as for all applications of the GDS. Refer to Section 5.5, of the *DCE 1.2.2 Administration Guide* for this.

Exit Values

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

Related Information

Refer to the **gdsditadm** reference page.

gdscp

Purpose is a command line interface you can use as an alternative to the Object Administration menu-interface and part of the Cache Administration menu-interface.

Synopsis **gdscp -c** {[*object*] *operation* [*dit-object-name*] [*-option value*]} [;]…

gdscp
filename

gdscp

Arguments

object The **gdscp** interface supports the following object types:

x500obj Represents the X.500 directory object. These are the objects in the DIT on which you will use **gdscp** to perform directory operations.

x500svc Represents the service controls that are passed in a directory operation (such as **search**, **list**, and so on). Refer to the *DCE 1.2.2 Application Development Guide* for more detailed explanation of how service controls are used by the Directory Service).

x500abbr Represents the abbreviations that are used by **gdscp** for attributes.

The *object* parameter is optional. If you do not specify an *object* parameter, **gdscp** defaults to the **x500obj** object.

operation Refer to the **x500obj, x500svc**, and **x500abbr** reference pages for descriptions of the operations supported by each object type.

dit-object-name
Refer to the **x500obj** reference page for a description of how to specify object names.

option Refer to the **x500obj, x500svc**, and **x500abbr** reference pages for descriptions of the options supported by each object type.

filename The filename of a user-defined script containing **gdscp** commands and/or any other valid Tools Command Language (TCL) commands.

Description

The **gdscp** program is a command line interface you can use as an alternative to the Object Administration menu interface and part of the Cache Administration menu interface of the GDS administration program. You can use **gdscp** to manipulate objects in the DIT. You can use **gdscp** to perform operations such as **create**, **modify**, **delete**, and **search** on DIT objects. You can perform SQL-like search operations. You can also manipulate objects in the DUA cache. The **gdscp** interface can be adapted to operate on any directory schema. You can specify DIT object names and attributes easily by using strings. The **gdscp** interface supports the TCL interface which allows the user to write scripts to perform repetitive tasks. Refer to the *DCE 1.2.2 Administration Guide—Core Components* for information about the basic concepts and features of TCL. You can direct the output of one command as an input to a subsequent command. The **gdscp** interface also supports history mechanism and line recall and editing.

The **gdscp** interface allows you to enter commands in the following modes:

- Interactive
- Command line

Interactive Mode

Enter the **gdscp** command without any arguments to activate interactive mode. At the **gdscp** prompt, enter a **gdscp** command. The **gdscp** interface executes the command, displays the result, and is ready to accept another command.

Command-Line Mode

Activate command-line mode from the system prompt by using one of the following methods:

- Enter the **gdscp** command with a filename of a script containing **gdscp** commands (and/or other valid TCL commands) as follows:

```
gdscp myown.tcl
```

- Enter the **gdscp** command with the **-c** option followed by **gdscp** commands. Enter multiple commands by separating them with a semicolon (;) as follows:

```
% gdscp -c "bind; show /C=de/O=sni/OU=ap11;quit"
```

Startup Scripts

When you invoke **gdscp**, the following script files are executed in the order shown:

[info library]/init.tcl
: Contains the standard TCL initialization scripts with definitions for the **unknown** and the **auto_load** commands; **[info library]** evaluates to the path where TCL is installed.

$gdscp_library/**init.gdscp**
: Contains the initialization scripts implementing the **gdscp** commands and tasks. The implementation sets the TCL variable *gdscp_library* to **/opt/dcelocal/gdscp** by default.

$HOME/.gdscprc
: Contains user customizations.

User-written Script Files
: These files are evaluated only if you specify the **TCL_PATH** environment variable. The files must end with the **.tcl** extension and should be in the directory specified by the environmental variable **TCL_PATH** . You can specify multiple paths by separating the pathnames with a colon (:). For example:

```
TCL_PATH=/home/nollman/develop/scripts:$HOME/scripts
```

The following example shows a sample TCL script file. The script performs a bind operation. If the bind is successful, it creates the objects **/C=de**, **/C=de/O=sni**, and so on.

```
#Example TCL Script file
#Perform a BIND operation
if {[catch bind] == 0} {
```

```
#Create the objects
create /C=de -attribute OCL=C
create /C=de/O=sni -attribute OCL=ORG
create /C=de/O=sni/OU=ap11 -attribute OCL=OU
create /C=de/O=sni/OU=ap11/CN=schmid -attribute OCL=ORP SN=schmid
}
```

The following sample TCL script file prints the value of the first returned result of a search operation:

```
#Example that demonstrates TCL commands
#Perform a BIND operation
bind
#Make a search request and store the result in variable "result"
set result [search /C=de/O=sni/OU=ap11 -filter CN='*' -subtree]
#Now access the first result item from the list and store it in
#the variable "first"
set first [lindex $result 0]
#Print the first result
puts -nonewline "The first result is : "
puts $first
```

The following sample TCL procedure deletes the immediate children of the specified object in the DIT.

```
#The "object" parameter is the object in the DIT
#whose immediate children are to be deleted
proc delete_children {object} {
foreach i [list $object] {
delete $i
}
}
```

Command Processing

The **gdscp** interface supports the TCL built-in commands apart from its own commands. If a command name is unknown to **gdscp**, it is considered to be an "unknown" command and is evaluated using the following algorithm:

- If the command is found in a TCL script file, **gdscp** executes the command.
- If the command exists as an executable UNIX program, **gdscp** executes the command. Therefore, you can invoke any UNIX command from the **gdscp** prompt (for example, **ls -l**).
- If you have invoked the command at the top level of the TCL shell and the command requests C-shell history substitution in one of the common forms (for example, !!, !<*number*> or \^*old* \^*new*), **gdscp** emulates the C-shell's history substitution.
- If you have invoked the command at the top level of the TCL shell and the command is a unique abbreviation for another command, **gdscp** invokes the command.

The **gdscp** interface makes use of two mechanisms to allow all object names, operation names, and options to be abbreviated to the shortest unique string.

The first is the standard TCL mechanism built into the TCL **unknown** command described in the previous paragraphs. This mechanism only works if the command is entered interactively. If the command is found in a script, abbreviation checking is not performed by the standard implementation of the **unknown** command. This is to discourage the practice of using abbreviations of commands in scripts.

The other mechanism used for abbreviations is built into the individual **gdscp** commands themselves. This allows the operation name to be abbreviated to the shortest unique operation supported for an object, and the options to be abbreviated to the shortest unique string representing an option supported by an object and operation. This form of abbreviation is always available, whether invoked interactively or using a script.

For example, consider the **modify** operation on the **x500svc** object:

```
x500svc modify -automaticcontinuation TRUE -sizelimit 100
```

In the abbreviated form, the same operation can be entered as follows:

```
x500s mod -a TRUE -si 100
```

Note: The **gdscp** interface renames the TCL **list** command to **llist** so that it does not conflict with the **gdscp list** command.

Terminating gdscp

The **exit** and **quit** commands terminate an interactive **gdscp** session and unbind from the current Directory session. Use the following command syntax:

exit *n*

quit *n*

Use the *n* argument to specify the exit value returned to the shell.

The following example terminates a session and returns an exit value of **56** to the shell:

```
exit 56
```

Line Recall and Editing

You can edit a line before it is sent to **gdscp** by typing certain control characters and escape sequences. To enter a control character, hold down **<Ctrl>** while pressing the appropriate character key. To enter an escape sequence, press **<Esc>** followed by one or more character keys. The escape sequences are case sensitive; the control characters are not.

You can enter an editing command anywhere on a line. In addition, you can press **<Return>** anywhere on the line.

You can specify a number [*n*] as a repeat count. To enter a repeat count, press **<Esc>** , a number, and the command you want to execute.

For example, **ESC 4 C-d** deletes the next four characters on a line.

Table 12-1 shows the control characters for line editing.

Table 12–1. Control Characters for Line Editing

Control Character	**Action Performed.**
C-A	Move to the beginning of the line.
C-B	Move left (backward) [*n*].
C-D	Delete the character [*n*].
C-E	Move to the end of the line.

Control Character	**Action Performed.**
C-F	Move right (forward) [*n*].
C-G	Ring the bell.
C-H	Delete character before the cursor [*n*].
C-I	Complete the filename (**Tab** key).
C-J	Done with line (**Return** key).
C-K	Kill to the end of the line (or column [*n*]).
C-L	Redisplay the line.
C-M	Done with line (alternate **Return** key).
C-N	Get next line from history [*n*].
C-P	Get previous line from history [*n*].
C-R	Search backward (or forward if [*n*]) through history for text; start line if text begins with an up arrow.
C-T	Transpose characters.
C-V	Insert next character even if it is an edit command.
C-W	Wipe to the mark.
C-XC-X	Exchange the current location and mark.
C-Y	Yank back the last killed text.
C-[	Start an escape sequence (<**ESC**> key).
C-]c	Move forward to the next character *c*.
C-?	Delete the character before the cursor [*n*].

Table 12-2 shows the escape sequences for line editing.

Table 12–2. Escape Sequences for Line Editing

Escape Sequence	Action Performed.
ESC C-H	Delete the previous word (**Backspace** key) [*n*].
ESC DEL	Delete the previous word (**Delete** key) [*n*].
ESC SPC	Set the mark (**SPACE BAR**); refer to **C-XC-X** and **C-Y** control characters in previous table.
ESC .	Get the last (or [*n*]th) word from the previous line.
ESC ?	Show possible completions.
ESC <	Move to the start of history.
ESC >	Move to the end of history.
ESC b	Move backward one word [*n*].
ESC d	Delete the word under the cursor [*n*].
ESC f	Move forward one word [*n*].
ESC l	Make the word lowercase [*n*].
ESC u	Make the word uppercase [*n*].
ESC y	Yank back the last killed text.
ESC w	Make area up to mark yankable.
ESC nn	Set repeat count to the number *nn*.

The **gdscp** interface also supports filename completion. For example, suppose the root directory has the following files in it:

- **bin**
- **vmunix**
- **core**
- **vmunix.old**

If you type **rm /v** and then press **<Tab>**, **gdscp** will finish off as much of the name as possible by adding **munix**. If the name is not unique, the terminal will sound a beep tone. If you press **<Esc>-?**, **gdscp** will display the two possible complete file names: **vmunix** and **vmunix.old**. If you respond by entering a period (.) and by pressing the **<Tab>** key, **gdscp** completes the filename for you.

Serviceability

Refer to Section 5.5 of the *DCE 1.2.2 Administration Guide* for more information on serviceability messaging.

EXAMPLES

The following TCL script examples demonstrate various **gdscp** operations.

Bind Command

The following script portion shows a **bind** command which performs an anonymous bind to a DSA for directory identifier **16**:

```
# Perform a bind operation to the specified DSA.
gdscp> x500obj bind -dirid 16 -dsa /C=de/O=sni/OU=ap11/CN=dsa/CN=dsa16
```

Create Command

The following script portion shows various examples of **create** commands:

```
# create the country object
gdscp> x500obj create /c=de -attribute ocl=c
# create the organization object
gdscp> x500obj create /c=de/o=sni -attribute ocl=org
# create the organizational-unit object
gdscp> x500obj create /c=de/o=sni/ou=nm123 -attribute ocl=ou
# Set the current working object variable
gdscp> set gdscp_cwo /c=de/o=sni/ou=nm123
/c=de/o=sni/ou=nm123
# create the common-name objects
gdscp> x500obj create cn=naik -attribute  { {ocl=orp} {sn=naik}
{tn=12345;369072;576268} {up=naik} }
```

```
gdscp> x500obj create cn=naik,ou=nm123 -attribute { {ocl=orp} {sn=naik}
{tn=99999} }
gdscp> x500obj create cn=mueller -attribute  { {ocl=orp} {sn=John}
{dsc=Software Professional} }
gdscp> x500obj create cn=miller -attribute  { {ocl=orp} {sn=Peter}
{dsc=Engineer} }
gdscp> x500obj create cn=maller -attribute  { {ocl=orp} {sn=Henry}
{dsc=Engineer} }
gdscp> x500obj create cn=meller -attribute  { {ocl=orp} {sn=John} }
# create the dsa name object
gdscp> x500obj create cn=dsa -attribute { {ocl=app} }
gdscp> x500obj create CN=dsa/CN=dsa-m10 -attribute { {ocl=dsa}
{PSA={TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30101'}} }
# create an aliased object for /c=de/o=sni/ou=nm123/cn=naik
gdscp> x500obj create /cn=na -attribute AON={/c=de/o=sni/ou=nm123/cn=naik}
OCL=ALI
```

Various Display and Modify Commands

The following script portion shows various examples of **gdscp** commands that either display the contents of the directory in various ways or allow the contents to be modified:

```
gdscp> x500obj show /cn=na
/C=de/O=sni/OU=nm123/CN=naik {OCL=ORP;PER;TOP} CN=naik SN=naik
{TN=12345;369072;576268}
{UP='\x6e\x61\x69\x6b'}
gdscp> x500svc modify -dontdereferencealias TRUE
gdscp> x500obj show /cn=na
/CN=na {OCL=ALI;TOP} AON={/C=de/O=sni/OU=nm123/CN=naik} CN=na
gdscp> x500svc modify -default
gdscp> x500obj delete /cn=na
# list under '/C=de'
gdscp> x500obj list /c=de
/C=de/O=sni
# list under '/'
gdscp> x500obj list /
/CN=Schema /C=de
# list under '/C=de/O=sni/OU=nm123'
```

gdscp> x500obj list -pretty

The output from the last command will be as follows, if all of the operations previously shown have been performed on the directory:

```
 1) /C=de/O=sni/OU=nm123/CN=naik
2) /C=de/O=sni/OU=nm123/CN=mueller
3) /C=de/O=sni/OU=nm123/CN=miller
4) /C=de/O=sni/OU=nm123/CN=maller
5) /C=de/O=sni/OU=nm123/CN=meller
6) /C=de/O=sni/OU=nm123/CN=dsa
7) /C=de/O=sni/OU=nm123/CN=naik,OU=nm123
```

In the following script portion, the command examples are resumed from above:

gdscp> x500obj show /C=de/O=sni/OU=nm123/CN=mueller -attribute SN DSC
/C=de/O=sni/OU=nm123/CN=mueller SN=John {DSC=Software Professional}
 # -> the object and attributes (only types) will be displayed
gdscp> x500obj show /C=de/O=sni/OU=nm123/CN=mueller -types
/C=de/O=sni/OU=nm123/CN=mueller OCL CN SN DSC

 # modify
 # -> the attribute telephone number will be removed
gdscp> x500obj modify /c=de/o=sni/ou=nm123/cn=naik -removeattr tn
gdscp> x500obj show /c=de/o=sni/ou=nm123/cn=naik
/C=de/O=sni/OU=nm123/CN=naik {OCL=ORP;PER;TOP} CN=naik SN=naik {UP='\x6e\x61\x69\x6b'}
 # modify
 # -> the attribute telephone number (with value) will be added
gdscp> x500obj modify /c=de/o=sni/ou=nm123/cn=naik -addattr
"tn=12345;369072;576268;9999"
gdscp> x500obj show /c=de/o=sni/ou=nm123/cn=naik -pretty

The output from the last command will be as follows, if all of the operations previously shown have been performed on the directory:

```
 1) /C=de/O=sni/OU=nm123/CN=naik
Object-Class     : Organizational-Person
: Person
: Top
Common-Name      : naik
Surname          : naik
Telephone-Number : 12345
: 369072
: 576268
: 9999
User-Password    : \x6e\x61\x69\x6b
```

In the following script portion, the command examples are resumed from above:

```
# modify
#y -> rdn of object /c=de/o=sni/ou=nm123/cn=meller will be
#changed to Loose
gdscp> x500obj modify /c=de/o=sni/ou=nm123/cn=meller -rdn cn=Loose
gdscp> x500obj show /c=de/o=sni/ou=nm123/cn=Loose
/C=de/O=sni/OU=nm123/CN=Loose {OCL=ORP;PER;TOP} CN=Loose SN=John
```

Compare and Search Commands

The following script portion shows various examples of **gdscp** commands that perform comparison and search operations on the contents of the directory:

```
# compare
# attribute telephone number 12345 has to be present
gdscp> x500obj compare /c=de/o=sni/ou=nm123/cn=naik -attribute tn=12345
```

The output from this command should be:

```
TRUE
```

if the commands previously shown have been performed on the directory.

The next command shows a search operation:

```
# search for all children of '/c=de/o=sni/ou=nm123'
# -> all objects and attributes with cn=naik and
#phonenumber=12345 or cn=dsa or cn=Loose will be found
gdscp> x500obj search /c=de/o=sni/ou=nm123 -filter {{cn=dsa ||  cn=Loose || \
(cn=naik && tn=12345)}} -allattr -onelevel -pretty
```

The output from this command should be as follows:

```
 1) /C=de/O=sni/OU=nm123/CN=naik
Object-Class     : Organizational-Person
: Person
: Top
Common-Name      : naik
Surname          : naik
Telephone-Number : 12345
: 369072
: 576268
: 9999
User-Password    : \x6e\x61\x69\x6b
2) /C=de/O=sni/OU=nm123/CN=Loose
Object-Class     : Organizational-Person
: Person
: Top
Common-Name      : Loose
Surname          : John
3) /C=de/O=sni/OU=nm123/CN=dsa
Object-Class     : Application-Process
: Top
Common-Name      : dsa
```

The next command shows a further search operation:

```
# search for whole subtree under '/c=de/o=sni/ou=nm123'
# -> all objects with cn=m* and the attributes sn and dsc
#will be found
gdscp> x500obj search /c=de/o=sni/ou=nm123 -filter {cn=m*} -attribute
{sn dsc} -subtree -pretty
```

The output from this command should be as follows:

```
 1) /C=de/O=sni/OU=nm123/CN=maller
Surname     : Henry
Description : Engineer
2) /C=de/O=sni/OU=nm123/CN=miller
Surname     : Peter
Description : Engineer
3) /C=de/O=sni/OU=nm123/CN=mueller
Surname     : John
Description : Software Professional
```

The following script portion shows one more example of a search operation:

```
# search for subtree (incl. base object) '/'
# -> all objects with cn~=mueller, OCL=ORP or OCL=REP
#and sn=(not)henry and dsc=* will be found
gdscp> search -filter {{(cn~=mueller) && ((OCL=ORP) || OCL=REP) && !
(sn=henry) && (dsc=*) }} -subtree
/C=de/O=sni/OU=nm123/CN=mueller /C=de/O=sni/OU=nm123/CN=miller
```

Delete Command

The following script portion shows various examples of **delete** operations:

```
# Remove the objects from the DIT
# remove the dsa object
gdscp> x500obj delete /C=de/O=sni/OU=nm123/CN=dsa/CN=dsa-m10
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=dsa
# remove the common name objects
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=Loose
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=maller
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=miller
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=mueller
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=naik,ou=nm123
```

```
gdscp> x500obj delete /c=de/o=sni/ou=nm123/cn=naik
# remove the organizational-unit object
gdscp> x500obj delete /c=de/o=sni/ou=nm123
# remove the organization object
gdscp> x500obj delete /c=de/o=sni
# remove the country object
gdscp> x500obj delete /c=de
```

Cache Commands

The following script portion shows various examples of operations on the cache:

```
# Bind to the cache
gdscp> x500obj bind -dirid 16 -cache
# Modify the service controls to perform operations on cache
gdscp> x500svc modify -duacache TRUE -usedsa FALSE -dontusecopy FALSE
gdscp> x500obj show /c=de/o=sni/ou=ap11/cn=dsa/cn=dsa16 -pretty
```

The output from this command should be as follows:

```
 1) /C=de/O=sni/OU=ap11/CN=dsa/CN=dsa16
Object-Class         : Directory-Service-Agent
: Application-Entity
: Top
Presentation-Address
Network-Address    : TCP/IP!internet=127.0.0.1+port=25016
T-Selector         : Server
```

Return Values

All **gdscp** commands return one of the following:

- A list of information requested by the user (such as the results of a search operation)
- NULL (indicating successful completion of an operation)
- An error message string

The **gdscp** interface uses the TCL native error handling facility to log additional error information. This additional information is stored in the two variables *errorInfo* and *errorCode*. The *errorInfo* variable contains a stack trace of each of the nested calls to the TCL interpreter when the error occurred. The *errorCode* variable is a TCL list containing two elements, **GDSCP** (identifying the **gdscp** program) and the numeric value of the error code. You can use the TCL **catch** command to determine the successful completion or failure of the various **gdscp** commands. Refer to **gdscp.h** header file for a description of the error codes.

To view the results in a structured format (for example, after a successful search operation), use the **-pretty** option. If you specify this option, the output of a command result is output in pages of 23 lines in length. You can scan through the output by using special **gdscp** scrolling commands:

n	View the *n* th page
-n	Skip *n* pages backward
+n	Skip *n* pages forward
$	View the last page
q	Quit
<SPACE>	Advance to the next page
<CR>	Advance one line

If you specify the **-pretty** option, the return value of the command will be NULL and not a TCL list.

Related Information

Refer to the *DCE 1.2.2 Administration Guide—Core Components* for information about the basic concepts and features of TCL.

Refer to the **x500abbr**, **x500obj**, and **x500svc** reference pages.

gdsdirinfo

Purpose

Displays information on GDS daemon processes and processes using GDS

Synopsis

gdsdirinfo [- **v** *debug_routing_specification_string*] [**-w** *routing_specification_string*] [**-P** *directory_name*]

Options

-v *debug_routing_specification_string*

Specifies where debug messages are routed to. For the syntax and semantics of the debug routing specification string refer to **svcroute(5)**. Only **BINFILE** may be specified for the ‘‘how’’ segment of the string.

-w *routing_specification_string*

Specifies where messages for exception handling should be routed to. For the syntax and semantics of the routing specification strings refer to **svcroute(5)**. The parameter may be replicated to specify different routings for each severity level.

-P *directory_name*

If this optional parameter is present, the logfile names given in the (debug) routing specification strings are assumed to be relative to the directory name (which contains the logfiles). The absolute pathnames of the logfiles are generated then by concatenation. Note the final ’/’ that separates the directory name and relative filename.

Description

The **gdsdirinfo** command is used to obtain information on all daemon processes running for GDS and on all current processes using GDS. All of the information is read by **gdsdirinfo** from the GDS-specific shared memory area and is written to **stdout**. First, a two-line header is printed, then the information specific to the different processes (one line per process) follows. The following information is displayed:

PROCTYPE

The process type. The following types can occur:

Monitor IPC-monitoring process

DUA-Cache
DUA-cache process

C-Stub C-stub process

S-Stub S-stub process

DSA DSA process

Dir-User Process using GDS (GDS Client)

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, in the case of activation and/or deactivation of the trace system). The processes are assigned IPC server IDs as follows:

1 DUA-cache process

2 **C-stub** process

5 IPC-monitoring process

11–30 **S-stub** processes

31–50 DSA-processes

If the process type is **Dir-User**, the IPCID displayed refers to the GDS IPC server (for example, **DUA-cache**, **C-stub**, **DSA**) with which this GDS client is associated.

Note that the following relationship exists between the directory identifier and the IPC server identifier for S-stub processes and DSA processes:

S-Stub Dir-ID = IPC-server-ID - 10

DSA Dir-ID = IPC-server-ID - 30

STATE Describes the state of the following:

- A GDS process during the startup phase. The following values are valid:

  W1 C-stub/S-stub tries to read its own PSAP address from DUA cache.

  W3 The DSA tries to read its own DSA name from the file.

  W4 The DSA tries to read the internal schema.

  W5 The DSA changes the schema object in the database.

- A GDS client. The values and their meanings are as follows:

  R1 IPC association between the GDS client and GDS server exists.

  R10 DAP/DSP association between the GDS client and GDS server exists.

If none of the specific states is associated with the process, a dash (-) is printed instead.

Examples

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
```

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

```
A shmget system call has failed (key = 1148635637,
access mode = 666, errno = 2).
```

Exit Values

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

gdsditadm

Purpose

Directory database administration program

Synopsis

gdsditadm [**-i** *directory ID*] [**-d** *dsa-dn*] [**-o** *operation*] [**-p** *password*] [**-u** *user*] [**-A** *authentication mechanism*] [*output*] [<*input*]

Options

-i *directory ID*
The directory ID. Valid values are 1 to 20.

-d *dsa-dn* DN of the bind DSA. If this option is not specified, a connection is set up to the default DSA. The following specifications are valid:

- The DN of the DSA; for example,

  /C=us/0=Smith Ltd./OU=dep.1/CN=DSA/CN=DSA1

- **SPECIFIC_DSA**: Mask 2 is displayed to select the DSA. (See the *DCE 1.2.2 Administration Guide*.)
- **CACHE**: connection to the DUA cache.

-o *operation* The administration function.

If the connection is to a DSA, valid values are as follows:

1 Object administration

2 Schema administration

3 Shadow administration

4 Subtree administration

If the connection is to the DUA cache, valid values are as follows:

1 Object administration

2 Cache update

If the option is not specified, the **gdsditadm** command outputs Mask 3. (See the *DCE 1.2.2 Administration Guide*.)

-p *password* User password (password of the administrator object) used in SIMPLE authentication.

-u *user* DN of the user (DN of the administrator object) used in SIMPLE authentication.

-A *authentication mechanism*
The authentication mechanism. Do not specify this option for anonymous authentication. Valid values are:

2 **Simple Unprotected**

5 **DCE Authentication**

25 **Simple Unprotected** and **DCE Authentication**

Arguments

output Name of the output file in batch mode. The output file is a simple filename (that is, it cannot contain directory pathnames). The named output file is created in the **/opt/***dcelocal***/var/adm/directory/gds/adm** directory. If no *output* parameter is specified in batch mode, then **/opt/***dcelocal***/var/adm/directory/gds/adm/tstfile** is created.

input Name of the input file in batch mode. For the *input* argument, any UNIX filename is valid. As it is a redirection, the file name is interpreted by the shell.

Description

The **gdsditadm** shell command is used to call the directory database administration. The options **-i**, **-d**, **-o**, **-p**, **-A**, and **-u** are only valid in dialog mode, and *output* and *input* are only used in batch mode.

Structure of the Input File in Batch Mode

An input file is required in batch mode. This file must include the mask entries in the same mask sequence as for dialog mode.

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

Comments can be used and must be enclosed between colons (:).

Comments that begin with ":*" (double quote, colon, asterisk, double quote) and end with "*:" are transferred from the input file to the output file.

Structure of the Output File in Batch Mode

The output file in batch mode contains the comments transferred from the input file and the program messages displayed in the masks. Error messages are indicated by the character string **ERROR**.

If the administration program is called with no option, it outputs the Logon Menu mask. (See the *DCE 1.2.2 Administration Guide*.)

Serviceability

Refer to Section 5.5 of the *DCE 1.2.2 Administration Guide* for information on serviceability messages.

Examples

An example of a **gdsditadm** batch mode call would be as follows:

gdsditadm *outfile* <*infile*

The remainder of this section shows examples of both the input and the output files in batch mode.

1. The following is an example of an input file:

```
:*****TEST 1 (Add Object) DSA OP=4*1****:
:*** sccsid = @(#)t1.1addobj 7.2 91/06/24 (K Sys AP 11) ***:
:Directory ID:1
:Authentication mechanism to be used:Simple unprotected
: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:Country
: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
```

2. The following is an example of an output file:

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

Exit Values

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

gdsipcinit

Purpose

Provides and removes IPC resources

Synopsis

gdsipcinit [**-l** *communication buffer size*] [**-s** *max. number of IPC-server entries*] [**-c** *max. number of IPC-client entries*] [**-u** *max. number of user credential entries*] [**-h** *max. number of heap buffer entries*] [**-d** *max. number of distributed command entries*] [**-r** *mxa. number of registration entries*] [**-R**][**-v** *debug_routing_specification_string*] [**-w** *routing_specification_string*] [**-P** *directory_name*]

OPTIONS

-l *communications buffer size*

Size of the communication buffer (in kilobytes), which must be within the range 1 <= *nn* <= **D23_MXSIZECBUF**. The default value is **D23_MXSIZECBUF / D23_MDEFREL**.

-s *max. number of IPC-server entries*

Maximum number of available server process entries, which must be within the range 1 <= *nn* <= **D23_MXSRVPROC**. The default value is **D23_MXSRVPROC / D23_MDEFREL**).

-c *max. number of IPC-client entries*

Maximum number of available client process entries, which must be within the range 1 <= *nn* <= **D23_MXCLTPROC**. The default value is **D23_MXCLTPROC / D23_MDEFREL**).

-u *max.number of user credential entries*

Maximum number of available user credential entries, which must be within the range 0 <= *nn* <= **D23_MXCRED**. The default value is **D23_MXCRED / D23_MDEFREL**).

-h *max. number of heap buffer entries*

Maximum number of available heap buffers, which must be within the range 0 <= *nn* <= **D23_MXHPBUF**. The default value is **D23_MXHPBUF / D23_MDEFREL**)

-d *max. number of distributed command entries*
Maximum number of available distributed command entries, which must be within the range 1 <= *nn* <= **D23_MXDISTCMD**. The default value is **D23_MXDISTCMD/D23_MDEFREL**).

Note: This option cannot be used in conjunction with the command options described above.

-r *max. number of registration entries*
Maximum number of available registration entries, which must be within the range 1 <= *nn* <= **D23_MXREGINFO**. The default value is **D23_MXREGINFO/D23_MDEFREL**.

-R Removes all IPC resources from the system.

-v *debug_routing_specification_string*
Specifies where debug messages are routed to. For the syntax and semantics of the debug routing specification string refer to **svcroute(5)**. Only **BINFILE** may be specified for the ‘‘how’’ segment of the string.

-w *routing_specification_string*
Specifies where messages for exception handling should be routed to. For the syntax and semantics of the routing specification strings refer to **svcroute(5)**. The parameter may be replicated to specify different routings for each severity level.

-P *directory_name*
If this optional parameter is present, the logfile names given in the (debug) routing specification strings are assumed to be relative to the directory name (which contains the logfiles). The absolute pathnames of the logfiles are generated then by concatenation. Note the final ’/’ that separates the directory name and relative filename.

Description

The **gdsipcinit** command provides or removes the IPC resources (shared memory, message queue, and semaphore) that are used by GDS to allow the communication between the different GDS components (such as **GDS-applications (DUA)**, **DUA cache**, **C-stub**, **S-stub**, and **DSA**).

The command is executed at activation time of GDS in advance of the execution of all other GDS components (daemons) and as the last command at deactivation

time. Because some areas of the IPC resource shared memory are configurable, the command accepts the options listed above.If an option is not specified, its default value is used.

Notes

To avoid inconsistent states in GDS, this command should be used outside the GDS administration very carefully. For example, call the command to remove IPC resources only if there is no GDS daemon still running.

Exit Values

If the command executes successfully, the exit value is 0 (zero); otherwise a self-explanatory error message is written to **stderr** and the exit value 1 is returned.

gdsipcstat

Purpose

Displays internal IPC information

Synopsis

gdsipcstat [**-v** *debug_routing_specification_string*] [**-w** *routing_specification_string*] [**-P** *directory_name*]

Options

-v *debug_routing_specification_string*

Specifies where debug messages are routed to. For the syntax and semantics of the debug routing specification string refer to **svcroute(5)**. Only **BINFILE** may be specified for the ‘‘how’’ segment of the string.

-w *routing_specification_string*

Specifies where messages for exception handling should be routed to. For the syntax and semantics of the routing specification strings refer to **svcroute(5)**. The parameter may be replicated to specify different routings for each severity level.

-P *directory_name*

If this optional parameter is present, the logfile names given in the (debug) routing specification strings are assumed to be relative to the directory name (which contains the logfiles). The absolute pathnames of the logfiles are generated then by concatenation. Note the final ’/’ that separates the directory name and relative filename.

Description

The **gdsipcstat** command is a tool that can be used by an administrator to localize and isolate GDS problems. It allows an administrator to print out internal IPC information, especially information from the shared memory segment (for example, resource bottlenecks or load problems). The functionality of the IPC package contained in GDS is implemented by means of the following system resources:

- Shared memory
- Message queue
- Semaphore and named pipe (FIFO)

The shared memory as an essential resource is used to exchange user data between IPC clients (for example, DUA and S-stub) and IPC servers (for example, DUA cache, C-stub, and DSA) and to store state information about all active GDS components.

The logical layout of the shared memory segment is as follows:

Table 12–3. Logical Layout of the Shared Memory Segment

General Area
Distributed Command Area *
Registration Area *
Server Area *
Client Area *
Communication Buffer Area *
User Credential Area *
Heap Area *

Note: Areas marked with an asterisk (*) indicate that the area is configurable.

The information displayed by the **gdsipcstat** command is explained in detail below.

General Area

The general area contains information about all other areas available in the shared memory segment. Information available in this area is always displayed by the **gdsipcstat** command.

Version number

The version number of the current shared memory layout. This number is updated every time a change is made in the layout of the shared memory. If an IPC operation, performed by any GDS component, fails with error number **D23_VERSIL** then the version of the IPC package linked to that component is incompatible to the shared memory layout actually in use.

Creation time
The date and time at which the shared memory segment was created (synonymous with the activation time of GDS).

Actual invoke-ID
The actual (global) **invoke-ID** that is assigned to the next occurring IPC operation. Through a global **invoke-ID** mechanism, IPC guarantees that an unambiguous **invoke-ID** is assigned to the IPC operations being active on the system (even if initiated by different processes).

Actual virtual process-ID
The actual **virtual process-ID** that is assigned to the next process attaching to IPC as a client or a server. Each process attached to IPC is associated with an unambiguous **virtual process-ID** used as an identifier during the exchange of IPC messages.

Maximum number of distributed command entries
The maximum number of entries available in the distributed command area. This value may be increased or decreased to actual requirements by a modification of the **-d** parameter found in the IPC configuration file *dce_local***/var/adm/directory/gds/conf/ipcconf.**

Actual number of distributed command entries
The number of entries in the distributed command area actually in use. The value is always 0 (zero) because the current implementation does not consider this field.

Maximum number of registration entries
The maximum number of entries available in the registration area. This value may be increased or decreased to actual requirements by modifying the **-r** parameter found in the IPC configuration file *dce_local* **/var/adm/directory/gds/conf/ipcconf**.

Actual number of registration entries
The number of entries in the registration area that are in use.

Maximum number of server entries
The maximum number of entries available in the server area. This value may be increased or decreased during configuration of GDS by increasing or decreasing the number of DSA server processes. Entries are occupied by IPC server processes like DUA cache, C-stub, DSA and S-stub (in the case of DSP).

Actual number of server entries
The number of entries in the server area that are actually in use.

Maximum number of client entries
The maximum number of entries available in the client area. This value may be increased/decreased during configuration of GDS by increasing/decreasing the number of client processes. Entries are occupied by IPC-client processes like GDS applications (DUA), C-stub (in the case of DUA-cache update or during initialization), S-stub (in the case of DAP or during initialization) and DSA (in the case of DSP).

Actual number of client entries
The number of entries in the client area that are actually in use.

Maximum number communication buffers
The maximum number of communication buffers available in the communication buffer area. One communication buffer is assigned to each client entry available in the client area (one-to-one relationship) because only IPC-clients can initiate IPC operations. During an IPC operation the communication buffer is shared between client and server.

Communication buffer size
The size of a communication buffer available in the communication buffer area. This size can be changed and adjusted to actual requirements (for example, to minimize IPC message fragmentation) by a modification of the **-l** parameter found in the IPC configuration file: *dce–local*/**var/adm/directory/gds/conf/ipcconf** The size must be specified in 1kB-blocks within the range of: 1 kilobyte <= *size* <= 16 kilobytes.

Maximum number of user credential entries
The maximum number of entries available in the user credential area. The number of entries is set equal to the number of client entries available in the client area during the configuration of GDS. An entry in this area is occupied by the DSA for every existing DAP/DSP association. The maximum number can be adjusted to actual requirements by a modification of the **u** parameter found in the IPC configuration file: *dce–local*/**var/adm/directory/gds/conf/ipcconf**

Note: If the configuration of GDS is changed by administration, the modified parameter value is overwritten.

Actual number of user credential entries
The number of entries in the user credential area that are actually in use.

Maximum number of heap buffers

The maximum number of heap buffers (each of size **D23_HPBFSZ**) available in the heap area. The number of buffers is set equal to the number of client entries available in the client area during the configuration of GDS. One or several (continuous) buffers are occupied (to hold the DN of a user (requester) of a DAP/DSP association) everytime the DSA adds the credentials of a user to the user credential area. The maximum number can be adjusted to actual requirements (for example, if very long DNs are in use) by a modification of the **h** parameter found in the IPC configuration file *dce-local*/**var/adm/directory/gds/conf/ipcconf**.

Note: If the configuration of GDS is changed by administration, the modified parameter value is overwritten.

State of heap buffers

There is a flag assigned to each heap buffer describing the state (0 = free, 1 = busy) of that buffer. The state information is displayed in units of 32 heap buffers. The lowest bit of the first unit corresponds to the state of the first buffer, the highest bit of the first unit corresponds to the state of the 32nd buffer, the lowest bit of the second unit corresponds to the state of the 33rd buffer, and so on.

For example:

```
0x3 0x4 0 0 0
```

means that buffer 1, 2 and 35 are busy and all other buffers are free.

Actual number of messages in message queue

The number of messages available in the message queue. If there is no IPC operation in progress then the message queue should be always empty. *Dead* messages (occurring, for example, if a GDS application is cancelled during a pending IPC operation) are normally removed by the IPC monitoring process. If these messages are not removed by the monitoring process, you must deactivate and reactivate GDS for cleanup.

Distributed command area offset

The decimal value offset from the beginning of the shared memory segment to the beginning of the distributed command area.

Registration area offset
The decimal value offset from the beginning of the shared memory segment to the beginning of the registration area.

Server area offset
The offset (decimal value) from the beginning of the shared memory segment to the beginning of the server area.

Client area offset
The offset (decimal value) from the beginning of the shared memory segment to the beginning of the client area.

Communication buffer area offset
The offset (decimal value) from the beginning of the shared memory segment to the beginning of the communication buffer area.

User credential area offset
The offset (decimal value) from the beginning of the shared memory segment to the beginning of the user credential area.

Heap area offset
The offset (decimal value) from the beginning of the shared memory segment to the beginning of the heap buffer area. The whole size of the shared memory segment used by IPC can be determined as follows:

heap area offset + (**D23_HPBFSZ** * *max. # of heap buffers*)

Distributed Command Area

The distributed command area (configurable in size) contains information about all distributed commands currently active. Each entry consists of the following:

Distributed command
The command in progress.

1	Close delta update logging file
2	Reopen delta update logging file
3	Disable database operations
4	Enable database operations
5	Restart server process
6	Disable GDS logging
7	Enable GDS logging

8	Close database files
9	Disable performance logging
10	Enable performance logging
11	Abandon request
12	Disable database modifications
13	Change serviceability logging levels

Process-ID of requester
The process identifier of the process that has initiated the distributed command.

IPC-(Server)-ID of the requested processes
The IPC ID of the recipients of the distributed command. If the distributed command is an abandon request, the ID of the requested process is the process ID.

1	DUA cache.
2	C-stub.
5	IPC monitoring process.
11–30	S-stub.
31–50	DSA.

Time stamp
The unambiguous time stamp value internally used to distinguish different distributed commands active at the same time.

Number of requested processes
The number of processes IPC has localized as recipients of the distributed command

Number of responding processes
The number of processes that have received and confirmed the distributed command. A distributed command is considered by IPC to be completed successfully if the number of responding processes is equal to the number of requested processes within the time limit (see below).

Time out value
The time value (in seconds) specifying the time frame in which an initiated distributed command has to be confirmed by all recipient processes.

Distributed command parameter value
The parameter value (string value of up to **D23_DPARSIZE** bytes) assigned to the distributed command and delivered by IPC to all recipients.

Registration Area

The registration area (configurable in size) contains information about all GDS components for which there is a need to be known by IPC (for example, to be able to receive distributed commands).

However, if the component is neither an IPC client nor an IPC server (like the IPC monitoring process) or not ready to do the predefined task (like the C-stub, S-stub, and the DSA during the initialization phase), the content of an entry is as follows:

Process type
Type of the registered process

2	C-stub
5	IPC monitoring process
11–30	S-stub process
31–50	DSA

Directory-ID
The directory identifier (1-20) with which the process is associated. If there is no association to any directory ID, then a dash (-) is printed.

Process-ID
The process identifier of the registered process.

State The state (**W** *xx*, **R** *xx*) of the registered process. Refer to the **gdsdirinfo** reference page for a detailed description about the valid states and their meanings. If no state is associated with the process, a dash (-) is printed.

Server Area

The server area (configurable in size) contains information about all active IPC server processes (for example, DUA cache, C-stub, and DSA).

Each entry consists of the following:

Server entry state

ES (entry state — bits 0 and 1)

00	Free entry
01	Entry occupied (attached)

OS (operational state — bits 2 through 5)

0001	IPC association exists
0010	IPC invoke pending + IPC-return buffer pending
0011	IPC invoke pending
0100	IPC request pending
0101	IPC result pending + IPC-return buffer pending
0110	IPC result pending

EM (own event mode — bits 6 and 7)

01	Synchronous event mode
10	Partially asynchronous event mode
11	Fully asynchronous event mode

PEM (event mode of corresponding IPC-client — bits 8 and 9)

01	Synchronous event mode
10	Partially asynchronous event mode
11	Fully asynchronous event mode

SET (server entry type — bits 10 and 11)

01	Primary server entry
10	Secondary server entry (temporarily used if several IPC operations are performed by one server process in parallel)

SPT (server process type — bits 12 and 13)

00	Unused (if own event mode is synchronous)

01 Single process server

10 Multiprocess server

UR (unload request — bits 14 and 15) (performed by DSAs only)

00 No unload request pending

01 Unload request pending

AR (abandon request - bit 16) (performed by DSAs only)

0 No abandon request pending

1 Abandon request pending

AD (abandon disabled/enabled - bit 17) (performed by DSAs only)

0 Abandon enabled

1 Abandon disabled

Unused (bits 18 through 31)
The state of the server entry is printed in a hexadecimal format (for example, 0x14C5). Additionally the entry state component or the operational state component is displayed as a text string.

Server entry substate
Not used (a dash (-) is always displayed)

Process-ID of server
The real/virtual process identifier of the process that has occupied the entry

IPC-(Server-)ID of server

1 DUA cache

2 C-stub

11–30 S-stub (the corresponding directory ID can be determined by subtracting 10 from the server ID)

31–50 DSA (the corresponding directory ID can be determined by subtracting 30 from the server ID)

IPC-association-ID
The IPC association ID of the corresponding IPC client. The number of the corresponding IPC client entry is equal to **IPC-association-ID** + 1 (displayed only if an IPC operation is in progress).

Process-ID of client
The real/virtual process identifier of the corresponding IPC client (displayed only if an IPC operation is in progress)

Global invoke-ID information
The **Invoke-ID information** of the IPC operation consisting of a **local invoke-ID**, **user-ID**, and a **context-ID** (displayed only if an IPC operation is in progress).

Client Area

The client area (configurable in size) contains information about all active IPC client processes (for example, GDS applications (DUA) and S-stub).

Each entry consists of the following:

Client entry state

ES (entry state — bits 0 and 1)

00	Free entry
01	Entry occupied (attached)

OS (operational state — bits 2 through 5)

0001	IPC association exists
0010	IPC invoke pending + IPC-return buffer pending
0011	IPC invoke pending
0100	IPC request pending
0101	IPC result pending + IPC-return buffer pending
0110	IPC result pending

EM (own event mode — bits 6 and 7)

01	Synchronous event mode
10	Partially asynchronous event mode

11 Fully asynchronous event mode

PEM (event mode of corresponding IPC-client — bits 8 and 9)

01 Synchronous event mode

10 Partially asynchronous event mode

11 Fully asynchronous event mode

Unused (bits 10 through 31)

Client entry substate
The state of the logical DAP/DSP association which corresponds to that IPC-association.

R1 No DAP/DSP association established

R10 DAP/DSP association established

Process-ID of client
The real/virtual process identifier of the process that has occupied the entry.

Requested server-ID
The server ID of the corresponding IPC server

1 DUA cache

2 C-stub

11–30 S-stub (only in the case of DSP)

31–50 DSA

Association-ID of corresponding server
The IPC-association-ID of the corresponding IPC server. The number of the corresponding IPC server entry is equal to **IPC-association-ID** + 1 (displayed only if an IPC operation is in progress).

Process-ID of corresponding server
The process identifier of the corresponding IPC server (displayed only if an IPC operation is in progress).

Global invoke-ID information
The Invoke-ID information of the IPC operation consisting of a **local invoke-ID**, **user-ID**, and a **context-ID** (displayed only if an IPC operation is in progress).

User Credential Area

The user credential area (configurable in size) contains the requester information about all existing DAP/DSP-associations.

Each entry consists of the following:

User-ID Describes the user ID:

IPC-association-ID of the IPC-client (DUA)
Used in the case of a local DAP association — see context-ID)

ROS-association-ID
Used in the case of a remote DAP/DSP-association (see **context-ID**)

Context-ID Describes the context in which the association to the DSA was established:

AC (Application context — bits 0 and 1)

01	Application context is DAP
10	Application context is DSP

AS (Abstract syntax — bits 2 and 3) (to be ignored)

01	Abstract syntax is ACSE
10	Abstract syntax is DAP
11	Abstract syntax is DSP

TS (Transfer syntax — bits 4 and 5) (to be ignored)

00	Transfer syntax is private (migration only)
01	Transfer syntax is ASN.1
10	Transfer syntax is private
11	Transfer syntax is ASN.1 (administration only)

AM (Access mode — bits 6 and 7)

01	Local access
10	Remote access (via S-stub)

RM (Remote-ID — bits 8 through 20)

11–30	Server ID of S-stub (remote access only)

Server-ID

The server ID of the DSA to which this entry is associated.

Requester information

The **Requester information** consists of the following:

requester type

0	Normal requester (objects with object class != **DSA**)
1	Privileged requester (DSA)

schema modification (1 byte)

0	No pending schema modification
1	Pending schema modification
2	Terminated schema modification
3	Broken schema modification
4	Schema modification unknown

authentication mechanism (2 bytes)

0	Anonymous
1	Simple authentication
2	DCE authentication
3	Strong authentication

trust level (4 bytes)

1	Read-only
2	Modify only
3	All (read and modify)

DN of the requester (1 – *n* bytes)

Empty string in the case of an anonymous requester

Examples

The following example contains sample **gdsipcstat** output with brief explanations of how to interpret some of the values.

```
GENERAL IPC-INFORMATIONS OF THE DIRECTORY INSTALLATION
Version number:                    V7.0
Creation time:                     Thu May  5 13:02:06 1994
Actual invoke-ID:                   3193
Actual virtual process-ID:           930
Max. # of dist. command entries:       4
Actual # of dist. command entries:     0
Max. # of registration entries:       16
Actual # of registration entries:      1
Max. # of server entries:            175
Actual # of server entries:            5
Max. # of client entries:            144
Actual # of client entries:            2
Max. # of communication buffers:     144
Communication buffer size (kB):        2
Max. # of user cred. entries:        144
Actual # of user cred. entries:        1
Max. # of buffers within heap:       144
State of the buffers within the heap:
0x0000003F    <--...00111111 = 6 heap buffers occupied
--- same ---
Actual # of messages in queue:             1
Dist. command area (offset):       +      332
Registration area (offset):        +      476
Server area (offset):              +      668
Client area (offset):              +     8368
Communication buffer area (offset):+    14704
User credential area (offset):     +   309616
Shared memory heap area (offset):  +   312496
```

```
Entire shared memory size:               314 kB
DISTRIBUTED COMMAND AREA INFORMATIONS:
Distributed command: 7   <-- 7 = enable GDS-logging
Requesting proc.-id: 18627
Requested Server-id/Process-id: 11 <-- 11 = S-stub (directory-ID 1)
Time-stamp:          64
No. of req. server:  1
No. of resp. server: 0
Time-out intervall:  30
Command parameter (size = 273):
0  2D 50 2F 6F 70 74 2F 64 63 65 6C 6F 63 61 6C 2F     |-P/opt/dcelocal/|
10  76 61 72 2F 64 69 72 65 63 74 6F 72 79 2F 67 64     |var/directory/gd|
20  73 2F 61 64 6D 2F 73 73 74 75 62 2F 20 09 2D 76     |s/adm/sstub/  -v|
30  67 64 73 3A 67 65 6E 65 72 61 6C 2E 31 2C 69 70     |gds:general.1,ip|
40  63 2E 32 2C 72 6F 73 2E 31 2C 63 6D 78 2E 31 3A     |c.2,ros.1,cmx.1:|
50  42 49 4E 46 49 4C 45 2E 32 2E 32 30 30 30 3A 4C     |BINFILE.2.2000:L|
60  4F 47 25 64 20 09 2D 77 46 41 54 41 4C 3A 54 45     |OG%d  -wFATAL:TE|
70  58 54 46 49 4C 45 2E 31 2E 31 30 30 3A 45 58 43     |XTFILE.1.100:EXC|
80  5F 46 25 64 20 20 2D 77 45 52 52 4F 52 3A 54 45     |_F%d  -wERROR:TE|
90  58 54 46 49 4C 45 2E 31 2E 31 30 30 3A 45 58 43     |XTFILE.1.100:EXC|
A0  5F 45 25 64 20 09 2D 77 4E 4F 54 49 43 45 3A 54     |_E%d  -wNOTICE:T|
B0  45 58 54 46 49 4C 45 2E 31 2E 31 30 30 3A 45 58     |EXTFILE.1.100:EX|
C0  43 5F 4E 25 64 20 2D 77 57 41 52 4E 49 4E 47 3A     |C_N%d -wWARNING:|
D0  54 45 58 54 46 49 4C 45 2E 31 2E 31 30 30 3A 45     |TEXTFILE.1.100:E|
E0  58 43 5F 57 25 64 20 09 2D 77 4E 4F 54 49 43 45     |XC_W%d  -wNOTICE|
F0  5F 56 45 52 42 4F 53 45 3A 54 45 58 54 46 49 4C     |_VERBOSE:TEXTFIL|
100  45 2E 31 2E 31 30 30 3A 45 58 43 5F 4E 56 25 64     |E.1.100:EXC_NV%d|
110  00                                                  |                |
IPC-REGISTRATION AREA INFORMATIONS:
1. Entry (Process-Type: 5  Directory-ID: 0 Process-ID: 16367 State:   -)
<-- 5 = IPC-monitoring process
IPC-SERVER AREA INFORMATIONS:
1. server entry:
Server state (0x745):    ATTACHED
<-- 0x745 = 0111 0100 0101
01 = ES is attached
0001 = OS is IPC-association
01 = EM is synchronous
11 = PEM of last operation
was fully asynchronous
```

```
01 = SET is primary entry
 Sub-state of server:       -
Process-ID of server:    real=16369/virt.=1
Server-ID of server:     1  <-- 1 = DUA-cache
2. server entry:
Server state (0x17c5):   ATTACHED
<-- 0x17C5 = 0001 0111 1100 0101
01 = ES is attached
0001 = OS is IPC-association
11 = EM is fully asynchr.
11 = PEM of last operation
was fully asynchronous
01 = SET is primary entry
01 = SPT is single process
server
Sub-state of server:       -
Process-ID of server:    real=16371/virt.=3
Server-ID of server:     2  <-- 2 = C-Stub
3. server entry:
Server state (0x14c5):   ATTACHED
<-- 0x14C5 = 0001 0100 1100 0101
01 = ES is attached
0001 = OS is IPC-association
11 = EM is fully asnchr.
00 = PEM is no last operation
01 = SET is primary entry
01 = SPT is single process
server
Sub-state of server:       -
Process-ID of server:    real=16373/virt.=5
Server-ID of server:     11  <-- 11 = S-Stub (directory-ID 1)
4. server entry:
Server state (0x745):  ATTACHED
<-- 0x745 = 0111 0100 0101
01 = ES is attached
0001 = OS is IPC-association
01 = EM is synchronous
11 = PEM of last operation
was fully asynchr.
01 = SET is primary entry
```

```
 Sub-state of server:       -
Process-ID of server:    real=16375/virt.=6
Server-ID of server:     31  <-- 31 = DSA (directory-ID 1)
5. server entry:
Server state (0x751):    REQUEST-PENDING
<-- 0x751 = 0111 0101 0001
01 = ES is attached
0100 = OS is request pending
01 = EM is synchronous
11 = PEM is fully asynchr.
01 = SET is primary entry
Sub-state of server:       -
Process-ID of server:    real=16378/virt.=7
Server-ID of server:     31  <-- 31 = DSA (directory-ID 1)
Association-ID of client: 0  <-- 0 = corresponding client
entry is 1 (DUA)
Process-ID of client:     real=17791/virt.=930
Global invoke-ID of operation in progress:
(local invoke-id=3191, usrid=0, context-id=0x69)
IPC-CLIENT AREA INFORMATONS:
1. client entry:
Client state(0x1d1):     REQUEST-PENDING
Sub-state of client:     R10   <-- R10 = existing out-
going DAP-
association
Process-ID of client:    real=17791/virt.=930
Requested server-ID:     31   <-- 31 = corresponding
server is DSA
(directory-ID 1)
Global invoke-ID of operation in progress:
(local invoke-id=3191, usrid=0, context-id=0x69)
2. client entry:
Client state(0x1351):     REQUEST-PENDING
Sub-state of client:      R1   <-- R1 = pending out-
going DSP-
association
 Process-ID of client:     real=16378/virt.=7
Requested server-ID:      11   <-- 11 = corresponding
server is S-stub
Process-ID of server:     real=16373/virt.=5
```

```
Global invoke-ID of operation in progress:
(local invoke-id=3192, usrid=1, context-id=0x4e)
USER CREDENTIAL AREA INFORMATONS:
User credential user-ID:     0  <-- 0 = IPC-association-ID
User credential context-ID:  0x41 <-- 0x41 = 0100 0001
01 = AC is DAP
00 = AS is ignored
00 = TS is ignored
01 = AM is local
User credential server-ID:   31
User credential (name):
0  00 00 00 02 00 00 00 03 38 35 2E 34 2E 36 04 64     |        85.4.6 d|
10  65 01 38 35 2E 34 2E 31 30 03 64 62 70 01 38 35     |e 85.4.10 dbp 85|
20  2E 34 2E 31 31 03 64 61 70 31 31 01 38 35 2E 34     |.4.11 dap11 85.4|
30  2E 33 03 61 64 6D 69 6E 00                          |.3 admin        |
<-- 00 = normal requester
<-- 00 = no pending schema
modification
<-- 00 02 = authentication
is simple
<-- 00 00 00 03 = trust level
is all
```

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

```
gdsipcstat: can’t get IPC-resources (errno = 2)
```

Exit Values

If the command executes successfully, the exit value 0 is returned; otherwise the return value is 1.

gdssetup

Purpose

Simplifies the process of configuring and initializing GDS

Synopsis

gdssetup [-*option*] gdssetup *filename* [-*option*]

Options

-h Displays help information.

-l *filename* Directs **gdssetup** to write logging and error information to the specified file. Error information is also duplicated in **stderr**. If you do not specify the **-l** option, no logging information is produced and error information is directed to **stderr**.

-o Gives permission to **gdssetup** to overwrite object attribute values where necessary.

If an object does not exist, **gdssetup** will perform the necessary operation (such as creating the object). If the object exists with an attribute that has to be modified and the limit of attributes has been reached, **gdssetup** will not add the attribute.

The **gdssetup** command will modify the attribute if the **-o** option was set when **gdssetup** was started. If more attributes are allowed, **gdssetup** will add the attribute. **gdssetup** will only modify the attribute if the **-o** option has been set.

Arguments

filename Specifies the pathname of a file if you use command line mode. The file must contain the appropriate input parameters and values as described in the next section. **gdssetup** processes the command as if you entered them interactively. If you do not specify a pathname, the **gdssetup** command assumes that you want to use Interactive mode.

Description

The **gdssetup** command provides the administrator with an interface to simplify the process of creating and initializing a directory configuration. The other method of creating and initializing a directory configuration requires that the administrator perform initialization steps by using the masks of the **gdsditadm** program either manually or by using batch file scripts (as described in Chapter 6.

Before starting the initialization of a DSA in an administrative domain, the administrator of the new DSA should contact the administrator of the initial DSA. The administrator of the new DSA must make sure that the administrator of the initial DSA creates a shadow of the entry at the root of the subtree that will hold the DN of the new DSA. This shadow entry is required so that the administrator of the new DSA can create objects under this root that make up the DN of the new DSA.

The administrator of the contact DSA should set the **Master-Knowledge** attribute of this entry equal to the DN of the new DSA. In addition, the administrator of the initial DSA should not set an ACL for the new DSA, but leave this for the administrator of the new DSA. Otherwise, the administrator of the new DSA will not have proper access rights to create the entry for the new DSA. Refer to Chapter 6 for more information on initializing a client/server system for an administrative domain.

Modes of Operation

You can run **gdssetup** in either of the following modes:

Interactive
: Prompts the administrator to enter the required information. The amount of information is a subset of the information required by using the **gdsditadm** program and is, therefore, substantially reduced. The administrator is not required to have a complete knowledge of the initialization steps. The **gdssetup** program performs initialization steps automatically which previously had to be specified explicitly by the administrator.

  Every interactive setup session is recorded in a file with the name taken from the environment variable **INTERACT_FILE**. If the environment variable is not present, **gdssetup** writes the information in **opt**/*dcelocal*/**var/adm/directory/gds/adm/gsu_setup_file**, using input file format syntax so that the file can be used as the input file in command-line mode.

  The file is used in the next interactive setup session. The recorded answers to the prompts are available by using the **editline** and **history** functions provided by DCE. The **history** output or the user input can

be edited and confirmed (by pressing <**Retrun**>). This feature can be very helpful because a failed setup often needs only a few changes.

Command Line
: Takes a filename as an input parameter that contains keywords and values similar to the prompts presented in interactive mode. This allows the administrator to create configurations with minor modifications. The input can also be read from **stdin**.

Interactive Mode

If you run **gdssetup** in interactive mode, you are presented with prompts that require the following types of data:

- Configuration
- Initialization
- Administration

For most configurations, it is sufficient to use the defaults.

The first set of prompts requests configuration data:

Dir-id The directory ID number. The default is **1**.

Config type (C/S)
: Enter the configuration type: **C** for client or **S** for client/server.

No-of-clients
: Enter the maximum number of clients that can access the directory at the same time. The default is **16**.

No-of-servers
: Enter the minimum number of server processes that can be active at the same time. The default is **2**.

Security method
: Valid values are:

  - **DCE**
  - **simple**
  - **simple**,**DCE**

  Enter **DCE** if you want to use DCE authentication as the security method; otherwise, enter **simple**. The default is **DCE**. If you want to use both security methods, enter **simple,DCE**.

The next set of prompts requests that you enter initialization data:

Client address
Enter the PSAP address of the client system. The default is **"TS=client,NA='TCP/IP!internet=127.0.0.1+port=30010'"**.

Local-DSA name
Enter the DN of the local DSA.

The first default DSA is also the local DSA (yes/no)
Enter **yes** if you want the local DSA to be the default DSA.

PSAP address
Enter the PSAP address of the local DSA.

Principal name
Enter the principal name of the local DSA if DCE authentication is the security method you specified earlier. This prompt does not appear if you specified **SIMPLE**.

DSA-password
Enter the password for the local DSA. **DSA-password** is mandatory for **SIMPLE** authentication, and optional otherwise.

Default-DSA name *n*
Enter the DN name of the default DSA. You can enter one or more default DSAs in the DUA cache. The DUA uses this information when an application wants to make a bind to a default DSA. The **gdssetup** command prompts you for the DN of each default DSA and its PSAP address.

PSAP address
Enter the PSAP address of default DSA *n*.

Contact-DSA name
Enter the DN of the contact DSA. If the new DSA holds the shadow of the schema object, the new DSA being configured is going to be in the same administrative domain as this contact DSA. Otherwise, the new DSA starts a new administrative domain. The contact DSA would be the source of the schema object if the **Use default schema** parameter is set to **no**. This DSA will be contacted to initiate all searches that are performed during the initialization process.

If this DSA name is not provided, it implies that the DSA being configured is the first DSA in the directory world. This is referred to as the *initial DSA*.

The **gdssetup** command prompts you for the PSAP address of the contact DSA (if you specified a contact DSA name).

PSAP address
Enter the PSAP address of the contact DSA.

Local schema is master (yes/no)
Enter **yes** if the local DSA is the master of the schema; otherwise, enter **no**.

Use default schema (yes/no)
Enter **yes** if you want to use the default schema; if you enter **no**, the local DSA takes the schema from the contact DSA.

The last set of prompts requests administration information:

Additional-DSA name *n*
Enter the names of additional DSAs to which an application can bind. To make a bind to one of these DSAs, the application must specify its DN (unlike the defaults DSAs where the DN is not required as a bind argument). This allows a client to directly bind to a DSA.

For each additional DSA name, **gdssetup** creates an object entry with **PSAP-ADDRESS** as its only attribute in the DUA cache. The **gdssetup** command prompts you for the DN of each additional DSA and its PSAP address. These DSA names will also be stored in the new local DSA so that they can be used for chaining and referral.

PSAP address
The PSAP address of additional DSA *n*.

The following table shows how you should set parameters for the five types of initializations described Chapter 6:

- Initial client/server system (that is, the first DSA in a distributed directory system) (**CL-SV/Init**)
- Client system (**CL**)
- Client/server system with nonGDS DSAs, or DSAs that do not constitute an administrative domain (**CL-SV/NoDom**)

- Client/server system, local DSA, and initial DSA constitute an administration domain and use the default schema (**CL-SV/Dom/Def**)
- Client/server system, local DSA, and initial DSA constitute an administration domain and do not use the default schema (**CL-SV/Dom/NoDef**)

Table 12–4.

Parameter	**CL**	**CL-SV INIT**	**CL-SV NoDom**	**CL-SV Dom Def**	**CL-SV Dom NoDef**
Dir-id	m	m	m	m	m
Client-address	m	m	m	m	m
Config-type (C/S)	C	S	S	S	S
Local-DSA name	-	m	m	m	m
Principal name	-	2	2	2	2
DSA password	-	1	1	1	1
Contact-DSA name	-	-	m	m	m
Local schema is master (yes/no)	—	yes	yes	no	no
Use default schema	-	yes	yes	yes	no
Additional-DSA name	o	o	o	o	o
Default-DSA name	o	o	o	o	o
No-of-clients	o	o	o	o	o
No-of-servers	—	o	o	o	o
Security method					
(DCE/simple/ simple,DCE)	—	o	o	o	o
The first default DSA is also the Local DSA (yes/no)	—	o	o	o	o

Note: The following notation is used for the preceding table:

—	parameter not allowed
m	mandatory parameter
o	optional parameter
1	mandatory for **SIMPLE** authentication;otherwise optional
2	mandatory if you specified **DCE** for the security method

Command-Line Mode

When you specify a filename when you run **gdssetup**, input is read from a file. You must create the input file by using a text editor and include the prompts that would be generated automatically in interactive mode by **gdssetup**.

The following sample input files for typical client (**client.init**) and client/server (**server.init**) configurations are included as part of the GDS software and are located in **/opt/** *dcelocal*/**var/adm/directory/gds/adm**.

You can edit these files and use them as a starting point for a client or client/server configuration. The following sample input file demonstrates a client configuration:

```
#########################################################################
#
#
# Sample input file for a client configuration with the tool
#
#    gdssetup
#
#
#
# NOTE: - the text before the angle bracket is fixed,
#       - the values have to be enclosed in quotation marks
#       - comment lines have to start with a '#' in the first column
#
#########################################################################
###
#
# Directory ID number - change it if you want to configure a Directory
#   ID different from 1
```

```
#
###
Dir-id> "1"
###
#
# Type of configuration - "C" means client configuration
#
###
Config-type(C/S)> "C"
###
#
#  PSAP address of client - you only should change the portnumber
#       (the last part of the address), if this portnumber is
#       already used
#       (use the netstat command to find out the used
#       portnumbers)
#
###
Client address> "TS=Client,NA='TCP/IP!internet=127.0.0.1+port=30010'"
###
#
# List of default DSAs - enter one or more default DSA to which the
#    client will perform the "bind to default DSA"
#    (the DUA will try to bind to the first given DSA,
#    only if this one is not available the DUA tries
#    to go to the next default DSA in the list and so on)
#         - remove the comments and replace the sample names
#    and PSAP addresses
#
###
#
#Default-DSA name<1>> "/C=DE/O=SNI/OU=NM12/CN=dsa/CN=dsa1"
#PSAP address <1>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30011'"
#Default-DSA name<2>> "/C=DE/O=SNI/OU=NM13/CN=dsa/CN=dsa2"
#PSAP address <2>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30021'"
#
###
#
# Additional DSAs - here you can enter additional DSAs to which you want
#      to be able to bind by only supplying the DSA-name and not
```

```
#      the PSAP-address
#    - remove the comments and replace the sample names
#      and PSAP addresses
#
###
#
#Additional-DSA name<1>> "/C=US/O=OSF/OU=DOC/CN=dsa/CN=dsa3"
#PSAP address <1>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30051'"
#Additional-DSA name<2>> "/C=US/O=IBM/OU=Dir/CN=dsa/CN=dsa4"
#PSAP address <2>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30061'"
#
```

The following sample input file demonstrates a client/server configuration:

```
##########################################################################
#
#
# Sample input file for a GDS server configuration with the tool
#
#    gdssetup
#
# COPYRIGHT ???
#
# NOTE: - the text before the angle bracket is fixed,
#       - the values have to be enclosed in quotation marks
#       - comment lines have to start with a '#' in the first column
#
#       Used abbreviations for the configuration types:
#  CL-SV/Init - Initial Client-Server System
#  CL-SV/NoDom - Client-Server System with Non-GDS DSAs, or
#      DSAs that do not constitute an
#      Administrative Domain
#  CL-SV/Dom/Def - Client-Server System where Local DSA and
#      Initial DSA do constitute an Administrative
#      Domain and use the Default Schema
#  CL-SV/Dom/NoDef - Client-Server System where Local DSA and
#      Initial DSA do constitute an Administrative
#      Domain and do not use the Default Schema
```

```
#
#############################################################################
###
#
# Directory ID number - change it if you want to configure a Directory
#    ID  different from 1
#
###
Dir-id> "1"
###
#
# Type of configuration - "S" means server configuration
#
###
Config-type(C/S)> "S"
###
#
# Number of clients - maximum number of clients which can attach to
#        IPC at the same time
#      - give a higher number than the default if you expect
#        a high usage frequency
#
###
No-of-clients> "16"
###
#
# Number of servers - enter the number of servers which will be
#        started when GDS is activated
#      - it should be more than one because this prevents the
#        server process from forking
#
###
No-of-servers> "2"
###
#
# Security method - enter here the security mechanism which the DSA
#      shall support
#      possible values: "DCE" for DCE authentication
#         "SIMPLE" for simple authentication
#           according X.509
```

```
#
###
Security method> "DCE"
###
#
# Principal name - enter the principal name of the DSA
#     (only if you have chosen DCE-security-method before !!!)
#   - the principal name format is:
#     /.../<cell name>/<principal name>
#
###
#
#Principal Name> "/.../dcecell_12/gdsdsa_6"
#
###
#
#  PSAP address of client - you only should change the portnumber
#       (the last part of the address), if this portnumber is
#       already used
#       (use the netstat command to find out the used
#        portnumbers)
#
###
Client address> "TS=Client,NA='TCP/IP!internet=127.0.0.1+port=30010'"
###
#
# Local DSA - enter name and PSAP-address of the local DSA
#       (remove the comments and replace the sample names
#        and PSAP addresses)
#
###
#
#Local-DSA name> "/C=DE/O=SNI/OU=NM11/CN=dsa/CN=dsa6"
#PSAP address> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30061'"
#
###
#
# Local DSA == Default DSA - change to "no" only if the local DSA
#        shall  NOT be in the list of default DSAs
#
```

```
###
The first default DSA is also the local DSA (yes/no)>"yes"
###
#
# DSA password - enter the password of the local DSA
#
###
#
#DSA password (max.10)> "value"
#
###
#
# Master of local schema - enter "yes" if the local DSA is the
#      master of the schema object (configuration types
#      CL-SV/Init and CL-SV/NoDom)
#    - otherwise enter "no" (configuration types
#      CL-SV/Dom/Def and CL-SV/Dom/NoDef)
#
###
#
#Local schema is master (yes/no)> "yes"
#
###
#
# Use default schema - enter "yes" if the default schema shall be used
#         (which is delivered with GDS and conformant to X.500)
#       - enter "no" if the schema shall be copied from the
#         contact DSA (only for configuration type
#         CL-SV/Dom/NoDef)
#
###
#
#Use default schema (yes/no)> "yes"
#
###
#
# Contact DSA - enter the name and the PSAP address of the DSA which
#  shall be contacted in all search requests and for providing the
#  schema information, if "Use default schema" is set to "yes"
#       - if this DSA is not given, it implies that the DSA being
```

```
#  configured is the first one in the Directory world
#  (configuration type CL-SV/Init)
#       - remove the comments and replace the sample names
#  and PSAP addresses
#
###
#
#Contact-DSA name> "/C=DE/O=SNI/OU=ADM/CN=dsa/CN=dsa1"
#PSAP address > "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30011'"
#
###
#
# List of default DSAs - enter one or more default DSA to which the
#    client will perform the "bind to default DSA"
#    (the DUA will try to bind to the first given DSA,
#    only if this one is not available the DUA tries
#    to go to the next default DSA in the list and so on)
#         - remove the comments and replace the sample names
#    and PSAP addresses
#         - you can't give it if you want to configure the
#    CL-SV/Init type
#
###
#
#Default-DSA name<1>> "/C=DE/O=SNI/OU=NM12/CN=dsa/CN=dsa2"
#PSAP address <1>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30021'"
#Default-DSA name<2>> "/C=DE/O=SNI/OU=NM13/CN=dsa/CN=dsa3"
#PSAP address <2>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30031'"
#
###
#
# Additional DSAs - here you can enter additional DSAs to which you want
#      to be able to bind by only supplying the DSA-name and not
#      the PSAP-address
#    - remove the comments and replace the sample names
#      and PSAP addresses
#    - you can't give it if you want to configure the CL-SV/Init
#      type
#
###
```

```
#
#Additional-DSA name<1>> "/C=US/O=OSF/OU=DOC/CN=dsa/CN=dsa4"
#PSAP address <1>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30041'"
#Additional-DSA name<2>> "/C=US/O=IBM/OU=Dir/CN=dsa/CN=dsa5"
#PSAP address <2>> "TS=Server,NA='TCP/IP!internet=127.0.0.1+port=30051'"
#
```

Serviceability

Refer to Section 5.5 of the *DCE 1.2.2 Administration Guide* for information on serviceability messages.

Logging And Error Information

Logging information consists of the following:

- The complete set of input parameters (including the results of a syntax check).
- All steps performed by **gdssetup**.
- All objects and attribute values that **gdssetup** reads and compares before modifying.
- All objects created and attribute values added by **gdssetup**.

Error information is reported when the following occur:

- A directory operation fails.
- An critical attribute value cannot be overwritten because you did not specify the **-o** option in the command line when you started **gdssetup**.
- ACLs prohibit the modification of an object (such as a DSA name).

Error information consists of the following:

- The operation that failed.
- The reason for the failure (if available).
- Suggested actions for the administrator to correct the problem (if available).

gdsstep

Purpose

GDS trace evaluation program

Synopsis

gdsstep [**-v**][**-t**][**-f**][**-h**][**-l** *level*] [**-s** *subcomp*] [**-i** *thread-id*] *filename*

Options

The following options determine how prologue information is displayed:

-v Displays complete prologue information and the content of every message **gdsstep** issues. This information consists of the following:

- Serviceability version
- Name of the program that has issued the message
- Thread ID
- Time stamp
- Debug level
- Message index
- Subcomponent of the message
- Source file and line number where the message was issued

-t Displays the time stamp of the time when the message was logged and the content of the message

-f Displays the name and line of the source file where the message was issued and the contents of the message

The following option control the output of data:

-h Displays raw data contained in message. Binary data is displayed in hexadecimal format. All other data is displayed as integers or strings. This option is provided for program developers who want to minimize the output of **gdsstep** and are able to read hexadecimal data.

The following options determine how messages are filtered:

-l *level* Sets the debug level up to the specified level for the messages that **gdsstep** issues. Valid values are **0** through **9**.**gdsstep** only displays messages up to the specified level.

-s *subcomp* Specifies the component about which **gdsstep** issues messages:

general	General GDS logging
IPC	Interprocess communications logging
apdu	Application program data units logging
asn1	Abstract Syntax Notation 1 logging
ros	Remote Operation Service logging
cmx	Transport Interface logging
pfm	Performance logging

-i *thread-ID* Specifies the thread about which **gdsstep** issues messages. Use the **-v** option to determine the *thread ID* number and to see all messages of all threads.

Arguments

filename Name of the log file that is to be evaluated.

Description

The **gdsstep** program evaluates a trace file containing a GDS trace. The result of the evaluation is in printable form.

Logging files that are generated by GDS applications can be evaluated and displayed at a later time with the **gdsstep** program. The information is intended to be self-explanatory. 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.

Files

See Chapter 5 for the names of the log files and the command that is used to evaluate them.

gdssysadm

Purpose

Directory system administration program

Synopsis

gdssysadm -f *function* [**-d** *directory ID*] [**-m** *conf.mode*] [**-c** *configuration type*] [**-C** *number of clients*] [**-s** *number of servers*] [**-o** *operation*] [**-M** *data medium type*] [**-n** *filename*] [**-k** [*password*]] [**-F** *formatting*] [**-v** *volume number*] [**-D**][**-p**][**-X**][**-A** *authentication mechanism*]

Options

-f *function*
The administration function to be executed. This option must always be specified as the first option. Valid values are:

c	Configure directory system
A	Activate directory installation
d	Deactivate directory installation
s	Save local directory data files
r	Restore directory data files saved
t	Deactivate trace system of the directory
T	Activate trace system of the directory
i	Display status information of the directory system

-d *directory ID*
The directory ID. Valid values are 1 to 20.

-m *conf.mode*
The configuration mode. Valid values are:

1	Enter configuration data
2	Delete configuration data
3	Display configuration data

4 Change configuration data

-c *configuration type*
The configuration type. Valid values are:

1 Client system

2 Client/server system

-C *number of clients*
The number of clients that can have access to the directory at the same time. Valid values are 1 to 256.

-s *number of servers*
The number of server processes to be activated. Valid values are 1 to 256.

-o *operation*
The backup operation code. Valid values are:

1 Initialize saving or restoring of data files

2 Write or read data files to or from data medium

3 End saving or restoring of data files

4 Determine number of data media required

-M *data medium type*
The type of media to be used. Valid values are:

0 Diskette

1 Tape

2 File

-n *filename*
The name of the file to be used for saving or restoring directory data. The filename is either an absolute or a relative filename. In the latter case, 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** (the default value of **TARPATH** is **/opt/***dcelocal* **/var/adm/directory/gds/adm**). This parameter is ignored if the **-M2** parameter is not set.

-k *password*
The password for protecting the directory system data files saved (maximum of 10 characters).

-F *formatting*
The code for the data medium formatting. Valid values are:

0	No formatting of data medium
1	Formatting of data medium

-v *volume number*
The volume number of the security data medium.

-D Suppresses output in a mask (for example, in the case of the option **Display of directory system status information** or **Display configuration data**, output is written to **stdout** and is used by another application). In the case of **Change configuration data**, any missing parameters are taken from the existing configuration; *directoryId* is therefore a mandatory parameter.

-p Suppresses error messages.

-X Turns on a trace of the administration procedure **gdssysadm**. The default is no trace.

-A *authentication mechanism*
The authentication mechanism. Do not specify this option for anonymous authentication. Valid values are:

2	Simple
5	DCE Authentication
25	Simple and DCE Authentication

Description

The **gdssysadm** command calls the GDS system administration program.

Combining Options

The following table shows which options can be specified in the command call, depending on the administration function to be executed. If one of these options is missing in the call, the **gdssysadm** command outputs the corresponding administration mask so that the user can enter the missing parameter interactively. The table also contains the number of the mask that the **gdssysadm** command outputs if one of the options is not specified. In this mask, all input fields where **gdssysadm** expects no input are omitted, because the parameter is already specified as an option.

Table 12–5.

Combining Options in the Command Call																	
Function	**Options**																**Mask**
	d	**m**	**c**	**C**	**s**	**u**	**o**	**M**	**n**	**k**	**F**	**v**	**D**	**p**	**X**	**A**	
c (configure)	d	m1	c1	C										p	X	A	3
c (configure)	d	m1	c2	C	s	u								p	X	A	3
c (configure)	d	m2												p	X		
c (configure)	d	m3											D	p	X		
c (configure)	d	m4		C									D	p	X	A	3
c (configure)	d	m4		C	s	u							D	p	X	A	3
A (activate)														p	X		
d (deactivate)														p	X		
s (save)	d							M[0-1]		k	F	v		p	X		5
s (save)	d							M2	n	k				p	X		5
s (save)	d						o1	M[0-2]		k				p	X		
s (save)	d						o2	M[0-1]			F	v		p	X		
s (save)	d						o2	M2	n					p	X		
s (save)	d						o3							p	X		
s (save)	d						o4							p	X		

r (restore)	d							M[0-1]		k				p	X		6
r (restore)	d							M2	n	k				p	X		6
r (restore)	d						o1	M[0-1]		k		v		p	X		
r (restore)	d						o1	M2	n	k				p	X		
r (restore)	d						o2	M[0-1]						p	X		
r (restore)	d						o2	M2	n					p	X		
r (restore)	d						o3										
r (restore)	d						o4										
t (do not log)														p	X		
T (log)														p	X		
i (display)													D	p	X		

Notes

The function code must always be specified as the first option. All the other options can be used in any sequence.

No blank spaces can be inserted between the ID and the value of an option.

Examples

The following command can be used to save the data files of a directory that uses the internal interface (as in dialog):

gdssysadm -fs [-d *directory ID***] [-M** *data medium type***] [-k[***password***]] [-F***formatting***] [-n** *filename***]**

The following calls are required in order to save the data files of a directory that uses a separate user interface. Note that the following examples are valid only for restoring from diskette or tape:

gdssysadm -fs -d*directory ID* **-o1-M** *data medium type* **-k[***password***] [-n***filename***]**

This call ensures that the DSA or DUA cache returns **TOO_BUSY** to each request.

gdssysadm -fs -d *directory ID* **-o4**

This call returns the number of data media to the **stdout** file.

It is necessary to perform the following commands once for each data medium required:

gdssysadm -fs -d*directory ID* **-o2-v** *volume number* **-M***data medium type* **-F***formatting*

This command indicates that the directory data is written to the media.

gdssysadm -fs -d*directory ID* **-o3**

This command ensures that the DSA or the DUA cache is accepting requests again.

The following command can be used to restore the data files of a directory that uses the internal interface (as in dialog):

gdssysadm -fr [-d *directory ID***] [-M** *data medium type***] [-k[***password***]] [-n***filename***]**

The following calls are required in order to restore the data files of a directory that uses a separate user interface. Note that the following examples are valid only for restoring from diskette or tape:

gdssysadm -fr -d*directory ID* **-o1-v** *volume number of first data medium*
-M *data medium type***-k[***password***]**

This call ensures that the DSA or DUA cache returns **TOO_BUSY** to each request.

gdssysadm -fr -d*directory ID* **-o4**

This call returns the number of data media to the **stdout** file.

It is necessary to perform the following commands once for each data medium required:

gdssysadm -fr -d*directory ID* **-o2-v** *volume number* **-M***data medium type*

This command specifies that the data medium is read.

gdssysadm -fr -d*directory ID* **-o3**

This command ensures that the DSA or the DUA cache is accepting requests again.

Exit Values

The following table lists the exit values and explains their meaning. The exit values *x*+1 through *x*+30 are used in case of activating through the command line.

Table 12–6.

Exit Values		
Values	**Type**	**Meaning**
1	Fatal error	Wrong syntax (*parameter*)
1	Fatal error	Illegal function code (*parameter*)
1	Fatal error	Parameter not allowed (*parameter*)

Exit Values		
Values	**Type**	**Meaning**
1	Error	The selected function can’t be executed
2	Error	The selected function can’t be executed
3	Error	The directory system is active
4	Error	The directory system is not configured
5	Error	The directory system is not active
6	Error	Configuration information does not exist
7	Error	Configuration data already exists
8	Error	Cannot send distributed command to directory system processes
9	Error	Wrong selection
10	Error	The directory system is still in use
11	Error	Cannot read media volume label
12	Error	Wrong media volume label
13	Error	This directory identifier is not configured
14	Error	Directory ID is not configured
15	Error	Invalid media volume configuration information

Exit Values		
Values	**Type**	**Meaning**
16	Error	Cannot format media volume
17	Error	Cannot write data to media volume
18	Error	Cannot read data from media volume
19	Error	Cannot read file list
21	Error	One file exceeds the size of the media volume
22	Error	Cannot find any files that can be saved
26	Warning	The restored database does not fit in the DSA
27	Warning	The restored database does not fit in the DUA cache
28	Error	Invalid input
30	Error	The process gdsipcinit cannot be started !
31	Error	The process gdsipcchk cannot be started !
32	Error	The process gdscache cannot be started !
33	Error	The process gdscstub cannot be started !
34	Error	The process gdssstub cannot be started !
35	Error	The process gdsdsa cannot be started !
36	Error	Can’t write data to file !

Exit Values		
Values	**Type**	**Meaning**
37	Error	Can’t read data from file !
38	Error	File doesn’t exist !
39	Error	Can’t read file list from file !
40	Error	Can’t execute administration !
41	Error	The selected function is executed by somebody else !
42	Warning	Conversion to the local string format fails !
43	Error	One of the fields could not be converted to the T61 string !
50	Warning	END key was pressed
52	Error	Do not know the path of the directory installation
53	Error	Problem in reading language file !
56	Error	DEL key was pressed
57	Error	Cannot get user name !

The following table lists the exit values which are used when activated through the command line.

Table 12–7.

Exit Values		
Values	**Type**	**Meaning**
x+1	Error	*process name* Invalid command line parameter
x+2	Error	*process name* Invalid directory ID
x+3	Error	*process name* Cannot change directory
x+4	Error	*process name* Cannot fork
x+9	Error	*process name* Key Error
x+10	Error	*process name* Invalid -r switch
x+11	Error	*process name* Invalid -p switch
x+12	Error	*process name* Invalid -n switch
x+13	Error	*process name* Invalid -b switch
x+14	Error	*process name* Invalid switch
x+15	Error	*process name* Invalid switch value
x+20	Error	*process name* Invalid count of maximum IPC associations
x+21	Error	*process name* Invalid count of maximum ROS associations
x+22	Error	*process name* No authentication mechanism given

Exit Values		
Values	**Type**	**Meaning**
x+23	Error	*process name* Invalid authentication mechanism
x+30	Error	*process name* Invalid interval time

Exit values *x*+1 through *x*+30 have the following values for *x*:

32 **gdsipcinit**

64 **gdsipcchk**

96 **gdscache**

128 **gdscstub**

160 **gdssstub**

192 **gdsdsa**

For example:

```
Exit code 194 -> = 192+2 gdsdsa: Error: invalid directory id
Exit code 52  -> = 32+20 gdsipcinit: Error: Invalid count of
maximum IPC associations
```

x500abbr

Purpose represents the abbreviations that are used by gdscp for attributes

Synopsis **x500abbr help** [*operation-name*] [**-verbose**]

x500abbr operations

x500abbr show [**-pretty**]

Operation-Name

help Displays the help text for the **x500abbr** object and its operations.

If **help** is invoked without any argument or a option, then it returns a one-line-per-operation help message with the following contents:

show Displays the attribute abbreviation.

help Displays help text for the x500abbr object and its operations.

operations Lists the operations that can be performed on x500abbr object.

If **help** is invoked with an operation name as the argument, it returns a one line per option help message.

For example:

```
%
x500abbr help show
-pretty     Displays the result in a structured format.
```

Finally, if **help** is invoked with the **-verbose** option, then a one paragraph description of the **x500abbr** object is returned. This command will return text explaining what the object represents and how to use it.

operations

Displays a list of operations that can be performed on the **x500abbr** object type.

show Displays abbreviations, the full names, and object identifiers of object classes and attributes. It also displays the abbreviations, the full names, and Structured Attribute Class Name of structured attribute classes.

The output is presented in one of two forms: a TCL list or formatted output resulting from the **-pretty** option. A TCL list is composed of the following elements:

{{*Object-Class-Abbr-list*} {*Attribute-Abbr-list*} {*Component-Abbr-list*}}

The first element, *Object-Class-Abbr-list,* is a TCL list and represents the object class abbreviations. Each item in this list is composed of the *object class abbreviation*, *full name*, and *object identifier* as follows:

```
{C Country 85.6.2} {ORG Organization 85.6.4} ...
```

The second element *Attribute-Abbr-list*, is a TCL list and represents the attribute abbreviations. Each item in this list is composed of the *attribute abbreviations, full name*, and *object identifier* as follows:

```
{C Country-Name 85.4.6} {CN Common-Name 85.4.3} ...
```

The third element *Component-Abbr-list* represents the abbreviations of the components of the structured attribute. Each item in this list is composed of the *structured attribute class name*, the *component abbreviation*, and the *full name* as follows:

```
{DS_C_TELEX_NBR {AB Answerback} {CC Country-Code}
{TN Telex-Number}} ...
```

If you specify the **-pretty** option, the output appears as follows:

```
Object Class Abbreviations:
C       Country                     85.6.2
ORG     Organization                85.6.4
ORP     Organizational-Person       85.6.7
ORR     Organizational-Role         85.6.8
OU      Organizational-Unit         85.6.5
Attribute Abbreviations:
C       Country-Name                85.4.6
CN      Common-Name                 85.4.3
O       Organization-Name           85.4.10
OCL     Object-Class                85.4.0
PA      Postal-Address              85.4.16
PSA     Presentation-Address        85.4.29
RA      Registered-Address          85.4.26
TXN     Telex-Number                85.4.21
Component Abbreviations for "DS_C_POSTAL_ADDRESS" class:
PA      Postal-Address
Component Abbreviations for "DS_C_TELEX_NBR" class:
AB      Answerback
CC      Country-Code
TN      Telex-Number
```

Options

-pretty Displays the output in tabular format.

-verbose Displays a description of the **x500abbr** object when used with the **help** command as follows:

```
%
x500abbr help -verbose
```

Arguments

operation-name
The operation about which help text is displayed

Description

The **x500abbr** object represents the attribute abbreviations that are used by the **gdscp** command line interface to the Directory Service. The **gdscp** command uses abbreviations to represent the GDS attributes so that you can specify attributes easily. The **x500abbr** object contains information about the abbreviation, full name, and object identifier of these attributes. It also contains information about the abbreviation, full name, and object identifier of the object classes. The **x500abbr** object holds information about the structured GDS attributes and their components. You can determine the abbreviation and other details of an attribute by performing a **show** operation on the **x500abbr** object.

Note: Make sure that any changes that are made to the schema are reflected in the XOM object information file: **xoischema**. The **gdscp** command uses the information in this file to create the lists of abbreviations displayed with the **x500abbr show** command. Refer to Appendix K for information on how and when to change the **xoischema** file.

Examples

The following example displays the help text for the **show** operation for the **x500abbr** object:

```
%
help show
-pretty     Displays the result in a structured format.
```

Return Values

All operations on the **x500abbr** object return one of the following:

- A list of information requested by the user
- NULL (indicating successful completion of an operation)
- An error message string

The **gdscp** command uses the TCL native error handling facility to log additional error information. This additional information is stored in the two variables: *errorInfo*

and *errorCode*. The *errorInfo* variable contains a stack trace of each of the nested calls to the TCL interpreter when the error occurred. The *errorCode* variable is a TCL list containing two elements: **GDSCP** (identifying the **gdscp** program) and the numeric value of the error code. You can use the TCL **catch** command to determine the successful completion or failure of the various **gdscp** commands. Refer to the **gdscp.h** header file for a description of the error codes.

Use the **-pretty** option to view the results in a structured format. If this option is specified, the output of a command result is output in pages of 23 lines in length.

If you specify the **-pretty** option, the return value of the command will be **NULL and not a TCL list.**

You can scan through the output by using special **gdscp** scrolling commands:

n	View the *n* th page
-n	Skip *n* pages backward
+n	Skip *n* pages forward
$	View the last page
q	Quit
SPACE KEY	Advance to the next page
CR	Advance one line

Related Information

Refer to the *DCE 1.2.2 Administration Guide—Core Components* for information about the basic concepts and features of TCL.

Refer to the **gdscp**, the **x500obj**, and the **x500svc** reference pages.

x500obj

Purpose

represents the X500 directory objects on which gdscp operations are performed

Synopsis

[x500obj]**bind** [**-cache**][**-dirid** *directory-ID*] [**-dsa** *dsa-name*] [**-psap** *psap-address*] [**-user** *username*] [**-password** *password*] [**-authentication** *authentication-method*]

[x500obj]**compare** [*dit-object-name*] **-attribute** *attribute-information*

[x500obj]**create** *dit-object-name* [**-attribute** *attribute-information*]…

[x500obj]**delete** *dit-object-name*

[x500obj]**help** [*operation-name*] [**-verbose**]

[x500obj]**list** [*dit-object-name*] [**-pretty**]

[x500obj]**modify** [*dit-object-name*] **-addattr** *attribute-information* [x500obj] modify [*dit-object-name*] {**-changeattr** *old-attribute-information new-attribute-information*}…

[x500obj]**modify** [*dit-object-name*] **-removeattr** *attribute-information*…

[x500obj]**modify** [*dit-object-name*] **-rdn** *attribute-information* [x500obj] operations

[x500obj]**search** [*dit-object-name*] [**-pretty**][**-types**][**-attribute** *attribute-type*]… [**-filter** *filter*] [**-allattr**][**-baseobject** | **-onelevel** | **-subtree**][**-noaliases**]

[x500obj]**show** [*dit-object-name*] [**-pretty**][**-types**][**-attribute** *attribute-type*]…

The **x500obj** argument is optional. If you do not include it, **gdscp** defaults to the **x500obj** object.

Operations

bind Binds **gdscp** to a directory server. All subsequent directory requests are directed to this server. You must execute a **bind** command before you attempt to perform any operation on the **x500obj** object.If a bind operation to the directory server has been performed earlier, this command makes an implicit unbind operation and then proceeds to

perform the new bind operation. You can perform a **bind** operation with user credentials.

If you do not specify an option, **gdscp** sends an anonymous bind request to the default DSA. The **-dsa** and **-psap** options are mutually exclusive. Use the **-dsa** option to specify the name of the DSA to which the bind is to be made. Use the **-psap** option to specify the PSAP address of the DSA to which the bind is to be made.

Use the **-dirid** option to specify the directory identifier to which the bind is to be made. If you do not specify a directory identifier, **gdscp** uses the default directory. When you use the **-dirid** option, you must also use either the **-dsa**, **-psap**, or **-cache** option.

If you use the **-cache** option, the bind refers to the DUA cache. You should not use the **-user**, **-password**, **-dsa**, **-psap**, or **-authentication** options with the **-cache** option.

The **-authentication** option takes the value **DCE** or **SIMPLE**. Specify the **-user** and **-password** options for simple authentication. If DCE authentication is required, you should not specify **-user** and **-password**. If you do not specify an authentication mechanism, then an anonymous bind is made.

compare Compares an attribute name and value with the attribute names and values of an object in the DIT. The **compare** command returns TRUE if the attribute name and value you specify matches the attribute name and value of the object you specify in the DIT. Otherwise, it returns FALSE.

create Creates an object entry in the DIT.

delete Removes an object entry from the DIT.

help Displays help text for the **x500obj** object.

If **help** is invoked without any argument or an option, then it returns a one-line-per-operation help message as shown below:

```
bind            Binds to the directory server.
compare         Checks if the object has the specified
attribute values.
create          Creates an object in the directory.
```

```
delete          Removes the specified object entry from
the DIT.
list            Lists all the children of the specified
object.
modify          Modifies the attribute values of an
object in the DIT.
search          Searches for objects (normally which
satisfy certain conditions).
show            Reads attributes of an object.
help            Displays help text for the x500obj
object and its operations.
operations      Lists the operations that can be
performed on x500obj object.
```

If **help** is invoked with an operation name as the argument, it returns a one-line-per-option help message.

For example:

```
% help show
-attribute  To request specific attribute values from the
DSA.
-types      To request only attribute types and no values
from the DSA.
-pretty     To display the result in a structured format.
```

Finally, if you invoke the **help** command with the **-verbose** option, it displays a one paragraph description of what the **x500obj** object represents and how to use it.

list Displays the children of the object you specify. For example:

```
list
```

lists the children of the current object. Assuming that the current object is **/C=de/O=sni/OU=ap11**, the returned TCL list would be as shown below:

```
/C=de/O=sni/OU=ap11/CN=mueller
/C=de/O=sni/OU=ap11/CN=schmid
```

In general, each item in the returned TCL list will contain a DN.

If you specify the **-pretty** option, the output would appear as follows:

```
1) /C=de/O=sni/OU=ap11/CN=mueller
2) /C=de/O=sni/OU=ap11/CN=schmid
```

Note: **gdscp** renames the TCL **list** command to **llist** so that it does not conflict with the **gdscp list** command.

modify Modifies the attribute values of an object in the DIT. The **modify** operation has four modes of operation depending on the option you specify. The options are:

-addattr Adds a new attribute or attribute values to an existing attribute. You should only specify one attribute type with one or more values. If the attribute you specify does not exist, the attribute is created.

-changeattr Modifies an existing value of an attribute. You can change more than one attribute value for more than one attribute type in a single execution.

-removeattr Deletes an existing attribute or attribute values from a specified object entry. If you specify only the attribute type, the command deletes the attribute from the object entry. You can delete more than one attribute type in a single execution.

-rdn Modifies the RDN of an object in the DIT.

The **modify** options are mutually exclusive, and only one of them must be used during a **modify** operation.

operations Displays a list of operations that can be performed on the **x500obj** object type.

search Starts a search from the object you specify and searches the DIT for entries that match a filter. The object you specify in the command is the base object. Use only one of the mandatory options: **-baseobject**,

-onelevel, or **-subtree** to limit the scope of the search to the base object, the next level of objects, or the subtree below the base object.

The search operation returns the name of all entries without attribute information by default. Use the **-attribute** option to request specific attributes. Use the **-allattr** option to request all attributes. Use the **-types** option to request attribute types only. The **-attribute**, **-allattr**, and **-types** options are mutually exclusive.

For example:

```
search /C=de/O=siemens/OU=dap11 -onelevel -attribute SN TN \
-filter {{(CN=s*) && (SN=*schmid*)}}
```

This command searches at the next level of **/C=de/O=siemens/OU=dap11** all the objects whose common name starts with **s** and whose surname contains **schmid**.

The returned TCL list would appear as follows:

```
{/C=de/O=siemens/OU=dap11/CN=schmid  {SN=Henry Schmid} \
TN=122345;789378}
```

In general, each item in the returned TCL list will contain the complete entry information of an object in the DIT. Furthermore, each entry information is also a TCL list of items, with the first item as the DN, and the subsequent items as attribute information.

If you specify the **-pretty** option, the output is formatted. If you do not specify the **-pretty** option, the output is presented as a TCL list.

show Reads an object entry and displays the values of the attribute types you specify. The **show** operation returns all the attributes and their values by default. Use the **-types** option to request attribute types only. Use the **-attribute** option to specify particular attributes. If you specify the **-pretty** option, the output is formatted.

If you do not specify the **-pretty** option, the output is presented as a TCL list. The first item in the list is the DN; subsequent items contain attribute information.

Arguments

dit-object-name

Specifies the *dit-object-name* of an **x500obj** object in one of the following formats:

- A complete object name relative to the root of the DIT

  The object name must start with the backslash (/).

  For example:

  "/C=de/O=dbp/OU=dap11/CN=mueller"

- An object name relative to the current position in the DIT

  To build a complete object name, **gdscp** appends the specified object name to the object name of the current position in the DIT. To specify the relative name, omit the beginning backslash (/) and start with the naming attribute abbreviation.

  For example:

  "CN=mueller,OU=sni"

  If the current object name is **/C=de/O=dbp/OU=dap11**, then **/C=de/O=dbp/OU=dap11/CN=mueller,OU=sni** is the target object name.

  The default current object is the root of the DIT. Specify the current object in the DIT by setting the global variable **gdscp_cwo**. **gdscp_cwo** is initialized with the specified value only if the object exists in the DIT. Otherwise, an error is returned.

  For example:

  ```
  set gdscp_cwo /C=de/O=dbp/OU=dap11
  ```

- No object name specified in the command

The current object is the target object.

Refer to the section entitled *Strings Representing a Distinguished Name* for complete information on the string syntax.

directory-ID Specifies the ID number of the Directory Service to which a **bind** operation is performed (1-20).

dsa-name Specifies the distinguished name of the DSA to which a **bind** operation is performed. It must be a pathname starting from **root** (for example, **/C=de/O=dbpOU=dap11CN=dsaCN=dsa-m1**. Refer to the section entitled *Strings Representing a Distinguished Name* for information on strings used to represent DNs.

psap-address

Specifies the Presentation Service Access Point Address (PSAP) of the DSA to which a **bind** operation is performed. Refer to Appendix H for a description of how to format a PSAP address. An example of a PSAP address follows:

"TS=Server, NA=TCP/IP!internet=127.0.0.1+port=21011"

username Specifies the distinguished name of the user on whose behalf the bind request is made. The *username* and *password* values form the user credentials for **SIMPLE** authentication. *username* requires a value with **DN** syntax. It must be a pathname starting from **root**. For example:

/C=de/O=sni/OU=dap11/CN=miller

Refer to the section entitled *Strings Representing a Distinguished Name* for information on strings used to represent DNs.

password Specifies the user password of the user on whose behalf the bind request had been made for **simple** authentication. For example:

schmid

or

\x6e\x61\x69\x6b

authentication-method
Specifies the authentication method used in a **bind** request. Valid values are

- **DCE**
- **SIMPLE** (requires a *username* and a *password*)

attribute-information
The combination of attribute type and value separated by an equal sign (=). For example:

```
SN=schmid or 85.4.4=schmid
```

Refer to the sections entitled *Strings Representing GDS Attribute Information* and *Strings Representing Structured GDS Attribute Information* for more information on string syntax.

attribute-type
Specifies the attribute type to be read from the DIT in **show** and **search** commands. Specify an attribute type by entering an attribute abbreviation or attribute object identifier. For example, **SN** (an attribute abbreviation) and **85.4.4** (an attribute object identifier) are synonymous.

operation-name
Specifies the operation for which help text is displayed.

filter Specifies a filter expression. Refer to the section entitled *Strings Representing Expressions* for complete information on string syntax for filters.

Options

Options contain a leading dash (-) and are a full word. Some options take a value that immediately follow the option. If the option allows multiple values, you can specify the values immediately after the option as separate arguments, as a single TCL list argument, or as a combination of both as shown in the following examples:

```
gdscp>x500obj create CN=schmid -attribute tn=1234 dsc=Engineer SN=schmid
gdscp>x500obj create CN=schmid -attribute {tn=1234 dsc=Engineer SN=schmid}
gdscp>x500obj create CN=schmid -attribute {tn=1234 dsc=Engineer} SN=schmid
```

If you use a space character in a value, you must enclose the entire value within two sets of braces as shown in the following examples:

```
gdscp>x500obj create CN=schmid -attribute {{dsc= Software Engineer}}
gdscp>x500obj create CN=schmid -attribute {{dsc= Software Engineer} tn=1234}
```

The **x500obj** object supports the following options:

-addattr Adds attributes in a **modify** operation.

-allattr Searches for all attributes.

-attribute Specifies the attribute information or attribute types.

-authentication
Specifies the authentication mechanism in a **bind** operation.

-baseobject Restricts the scope of a search to base object only.

-cache Binds to the DUA cache.

-changeattr Changes attributes in a **modify** operation.

-dirid Specifies a directory identifier.

-dsa Specifies a DSA name.

-filter Uses the specified filter in a search operation.

-noaliases Specifies that aliases are not to be dereferenced in a **search** operation.

-onelevel Restricts the scope of a search to all children of the base object.

-password Specifies a user password.

-pretty Presents a result in structured format.

-psap Specifies a PSAP address.

-rdn Modifies a leaf RDN in a **modify** operation.

-removeattr Deletes attributes in a **modify** operation.

-subtree Specifies the scope of a search as the whole subtree of the base object (including the base object).

-types Restricts a **search** or **show** request to attribute types and no values.

-user Specifies a user name.

-verbose Displays a description of the **x500obj** object when used with the **help** command.

Description

The **x500obj** object represents X.500 directory objects in the DIT on which **gdscp** commands perform Directory operations (such as **create**, **modify**, and **delete**). You can perform SQL-like **search** operations. You can also manipulate the objects in the DUA cache.

You can specify the DIT object names and attributes easily by using strings. The **x500obj** object type always remembers the current object in the DIT so that you do not have to specify the DIT object name with every operation or can specify the DIT object relative to the current object. You can also specify the attribute values as **T.61** strings or strings in Local Code (ISO8859-1).

Support for T.61 Strings

The **gdscp** command assumes by default that strings are specified with the Local Code Set (ISO8859-1). This implies that you can specify a value such as **CN=Müller**. However, you can set the TCL global variable **gdscp_t61** to TRUE so that **gdscp** interprets strings to be of type **T.61**. If you set **gdscp_t61** to FALSE or do not define it, **gdscp** assumes Local Strings.

Specifying String Syntax with gdscp Commands

You can specify DIT object names and attributes using one of the following syntax types:

- Strings representing GDS attribute information
- Strings representing structured GDS attribute information
- Strings representing a distinguished name
- Strings representing expressions

Strings Representing GDS Attribute Information

These strings are used to associate the attributes with their values. They are of the form:

<Attribute Type> = <Attribute Value>

The attribute types can either be specified as abbreviations or object identifier strings. An object identifier string is defined as a series of digits separated by the dot character. If attribute abbreviations are used, they are case insensitive. For example:

cn=schmid or **85.4.3=schmid**

Object class values can also be specified as an abbreviation string. For example, an object class for **Residential Person** can be specified as **OCL=REP** or **OCL='\x55\x06\x0A'**.

All leading and trailing whitespace (surrounding the attribute type, the equal (=) character, and the attribute value) is ignored.

The following are the reserved characters for such strings:

' These can be used to enclose the attribute values. If this character is used, all other reserved characters within the quoted string except the \ character are not interpreted (for example, **cn='henry mueller'**).

; To separate multiple values of a recurring attribute. All leading and trailing whitespace surrounding the semicolon (;) is ignored. For example:

TN=899898;979779

= To associate the attribute with its value.

\x *nn* Used to specify hexadecimal data. After the **\x**, the next two characters are read as the hexadecimal value.

\ Used to escape any of the reserved characters above.

Strings Representing Structured GDS Attribute Information

These strings are used to associate the structured attribute and its components with their values. They are of the form:

<Structured Attribute Type> = {<Comp1 = Value>, <Comp2 = Value>, ..}

The structured attribute type can either be specified as abbreviations or object identifier strings. An object identifier string is defined as a series of digits separated by the dot character. If attribute abbreviations are used, they are case insensitive. *Comp1*, *Comp2*, and so on, are the components of the structured attribute. They should be specified as abbreviations. For example:

TXN={TN=977999, CC=345, AB=8444}

Specify the recurring values for structured attributes using a semicolon (;) as follows:

TXN={TN=977999, CC=345, AB=8444};{TN=123444,CC=345, AB=8444}x

Specify the recurring values for the components as follows:

TXN={TN=977999; 274424, CC=345, AB=8444}

If any of the components are further structured, enclose them within braces as follows:

FTN={PA={FR=1,TD=1}, PN=67899}

All leading and trailing whitespace surrounding the structured attribute type, the component abbreviation, the equal (=) character, the open brace ({) , the comma (,), and the closed brace (}) is ignored.

Specify attributes and components with distinguished name syntax as follows:

```
AON={/c=de/o=sni/ou=ap11/cn=mueller}
ACL={MPUB={INT=0, USR={/c=de/o=sni/cn=mueller, sn=schmid}}}
```

Specify the object class value as an abbreviation string as follows:

```
SG={OCL=REP} or SG={OCL='\x55\x06\x0A'}
```

Attributes of type Presentation Address are handled by using the PSAP macro utility. Specify the value for such an attribute as follows:

```
PSA={TS=Server, NA='TCP/IP!internet=127.0.0.1+port=12345'}
```

Refer to Appendix H for further information on network address format.

The following are the reserved characters for strings with structured attribute information:

' Encloses the attribute values. If the **'**character is used, all other reserved characters within the quoted string except the \ character are not interpreted. For example:

```
cn='henry mueller'
```

/ Specifies an attribute value with DN Syntax. For example:

```
AON = {/c=de/o=sni/ou=ap22/cn=mayer}
```

{ Indicates the start of a structured attribute value block.

} Indicates the end of a structured attribute value block.

, Separates the components of a structured attribute. For example:

```
TN=977999, CC=345, AB=8444
```

It can also be used to specify multiple AVAs in the case of attributes with DN syntax.

; Separates multiple values of a recurring attribute or the recurring components of the structured attribute. All leading and trailing whitespace surrounding the semicolon (;) is ignored. For example:

TXN={TN=977999,CC=345,AB=8444};{TN=53533,CC=242,AB=44242}

= Associates the components with their value and the components to the structured attribute.

\x *nn* Specifies hexadecimal data. After the **\x**, the next two characters are read as the hexadecimal value.

\ Escapes any of the reserved characters above.

Strings Representing a Distinguished Name

Use these strings to represent the DN of the object in the following format:

/Attribute Type = Naming Attribute Value ....
or
/Attribute Value/Attribute Value ....

Specify the attribute types as abbreviations or object identifier strings. An object identifier string is defined as a series of digits separated by a dot (.). Attribute abbreviations, are case insensitive. Specify multiple AVAs by separating the naming attribute values with a comma (,).

You can specify the first RDN as the DCE global root string **/...**, which is a sequence of the backslash (/) character followed by three dots (...). Some examples are shown below:

/c=de/o=sni/ou=ap11, l=munich/85.4.3=schmid
/c=us/o=osf/ou=abc/subsystems/server/xyz
/.../c=us/o=osf/ou=abc/subsystems/server/xyz

The first nonspace character should always be the backslash (/) character. All leading and trailing whitespace surrounding the backslash (/), attribute type, the equal (=) character, and the attribute value) is ignored.

The following are the reserved characters:

' Enclose the naming attribute values. If you use this character, all other reserved characters within the quoted string except the \ character are not interpreted. For example:

```
cn='henry mueller'
```

/ Used as a delimiter between RDNs.

, Specifies multiple AVAs. All leading and trailing whitespace surrounding the comma (,) character is ignored. For example:

```
/c=de/o=dbp/ou=dap11/cn=schmid, ou=ap11
```

= Associates the object with its naming attribute value.

\xnn Specifies hexadecimal data. After the **\x**, the next two characters are read as the hexadecimal value.

\ Escapes any of the reserved characters above.

Strings Representing Expressions

These strings are used to specify an SQL-like expression in a **search** operation. The following example searches for any **Organizational-Person** (**ORP**) or **Residential-Person** (**REP**) whose name approximately matches **schmid** and whose surname is not **ronnie**:

```
(CN~=schmid) && (OCL=ORP || OCL=REP) && !(SN=ronnie)
```

You can use object identifiers instead of attribute type abbreviations. The object identifier string is a series of numbers separated by a dot (.)

All leading and trailing whitespace surrounding the attribute types, the operators, and the attribute values is ignored.

Additionally the presence of an attribute can also be tested in either of the following ways:

c = de && cn
c = de && cn = *

The following are the reserved characters:

' Indicates the start/end of an attribute value string. You should use it when spaces are part of the data. If this character is used, all other reserved characters within the quoted string except the \ character are not interpreted. For example:

OU=sni && cn='Henri Mueller' && tn=89989

/ Specifies an attribute value with DN Syntax. For example:

```
AON = {/c=de/o=sni/ou=ap22/cn=mayer}
```

= Associates the attribute with its value.

&& Logically ANDs two conditions.

|| Logically ORs two conditions.

! Logically negates a condition.

~= Specifies phonetic matching during a search.

< Matches values less than a specified value.

> Matches values greater than a specified value.

>= Matches values greater than or equal to a specified value.

<= Matches values less than or equal to a specified value.

*****
: Specifies substrings during a search.

(
: Used for nesting of filters.

)
: Used for nesting of filters.

{
: Indicates the start of a structured attribute value block.

}
: Indicates the end of a structured attribute value block.

,
: Separates the components of a structured attribute. For example:

```
TN=977999, CC=345, AB=8444
```

You can also use it to specify multiple AVAs in the case of attributes with DN syntax.

\xnn
: Specifies hexadecimal data. After the **\x**, the next two characters are read as the hexadecimal value.

****
: Escapes any of the reserved characters above.

While evaluating an expression during **search** operations, the following precedence of operators prevail:

( ) ! && ||

The **(** and **)**have the highest precedence and **||** has the lowest.

The gdscp Reserved Characters Interpreted by TCL

Some of the reserved characters used by **gdscp** (namely, braces ({}), the backslash (\) and semicolon (;)) are also interpreted by the TCL interpreter. To avoid the interpretation of these characters by the TCL interpreter, the user can surround the argument with braces ({}) or use the TCL reserved character, slash (\), to escape these characters.

For example, if the value of an attribute is **\x35\x36\x37**, you can use

```
modify -addattr {dsc=\x35\x36\x37}   or
modify -addattr dsc=\\x35\\x36\\x37
```

Filters

You can create complex filters by using expressions described in the previous section for search operations on an **x500obj** object. Construct filter expressions from the following types of items:

Complete matching (=)

Use the format: *attribute type = attribute-value*. For example:

```
SN=schmid
```

(which is the same as: 85.4.4=schmid).

Substring matching (=)

Substring match at the end

Use the format: *attribute-type = *partial-attribute-value*. For example:

```
SN = *id
```

Substring match in the beginning

Use the format: *attribute-type = partial-attribute-value**. For example:

```
SN = sc*
```

Substring match anywhere

Use the format: *attribute-type = *partial-attribute-value**. For example:

```
SN = *hm*
```

Greater than or equal to matching (>=)

Use the format: *attribute-type >= attribute-value*. For example:

CN >= S

Less than or equal to matching (<=)
Use the format: *attribute-type* <= *attribute-value*. For example:

CN <= T

Present matching (=)
Use the format: *attribute-type* = *. For example:

DSC = *

Approximate match based on phonetic extension (~=)
Use the format: *attribute-type* ~= *attribute-value*. For example:

L ~= muenchen

Use the following operators to combine filter items to generate filter expressions:

- && (AND)
- || (OR)
- ! (NOT)

For example, the **search** expression:

CN ~= mueller && ((OCL = ORP) || (OCL = REP)) && !(SN = henry) && (HBY = *)

looks for an object that satisfies the following filter criteria:

1. *Common-Name* (**CN**) matches approximately to "mueller"
2. Object class is **Organization-Person** (**ORP**) or **Residential-Person** (**REP**)
3. **Surname** (**SN**) is not "henry"
4. Object has **Hobby** (**HBY**) attribute

Examples

bind The following example demonstrates a **bind** command which performs a bind with user credentials to the specified DSA for directory identifier 2:

```
bind -dirid 2 -dsa /C=de/O=sni/OU=ap11/CN=dsa/CN=dsa15 \
-user /C=de/O=siemens/OU=dap11/CN=miller -password xxx \
-authentication simple
```

compare The following example returns TRUE if the current object has **schmid** as one of the values of the **SN** attribute:

```
compare -attribute SN=schmid
```

create The following example creates an object (the current object is **/C=de/O=sni/OU=ap11**) **/C=de/O=sni/OU=ap11/CN=mueller** with the specified attributes:

```
create CN=mueller -attribute "OCL=ORP;PER" SN=schmid
```

delete The following example deletes the child of the current object which has **CN=mueller**. **CN=mueller** is the object name relative to the current position:

```
delete CN=mueller
```

help The following example displays help text for the **bind** operation:

```
help show
-attribute  To request specific attribute values from
the DSA.
-types      To request only attribute types and no values
from the DSA.
```

```
-pretty     To display the result in a structured format.
```

list The following example displays the children of the current object, **/C=de/O=siemens/OU=ap11**:

```
list
/C=de/O=siemens/OU=ap11/CN=mueller
/C=de/O=siemens/OU=ap11/CN=schmid
```

If you specify the **-pretty** option, the output appears as follows:

```
1) /C=de/O=siemens/OU=ap11/CN=mueller
2) /C=de/O=siemens/OU=ap11/CN=schmid
```

modify The following example adds the attribute **Description** (if it does not already exist) with the values **xxx** and **yyy**. Otherwise, it adds the values to the existing attribute:

```
modify /C=de/O=siemens/OU=dap11/CN=mueller \
-addattr "DSC=xxx;yyy"
```

The following example changes the value of the **SN** attribute from **xxx** to **yyy** and changes the value of the **DSC** attribute from **aaa** to **bbb**:

```
modify CN=mueller -changeattr SN=xxx SN=yyy \
-changeattr DSC=aaa DSC=bbb
```

The following example deletes the two values of the **Description** (**DSC**) attribute and deletes the attributes **Locality** (**L**) and **Business-Category (BC**) from the object entry:

```
modify CN=mueller -removeattr "DSC=xxx;yyy" L BC
```

The following example changes the specified object (assuming the current object to be **/C=de/O=siemens/OU=dap11**) to **/C=de/O=siemens/OU=dap11/CN=mueller,OU=sni**:

```
modify CN=mueller -rdn "CN=mueller,OU=sni"
```

operations The following example shows the output of an **operations** operation:

```
operations
bind compare create delete list modify search show
help operations
```

search The following example searches at the next level in the DIT from **/C=de/O=siemens/OU=dap11** all the objects with **Common Name** starting with **S** and **Surname** containing **schmid**:

```
search /C=de/O=siemens/OU=dap11 -onelevel -attribute SN TN -filter
{{(CN=s*) && (SN=*schmid*)}}
```

The returned TCL list appears as follows:

```
{/C=de/O=siemens/OU=dap11/CN=schmid {SN=Henry Schmid} \
TN=122345;789378}
```

If you specify the **-pretty** option, the output is formatted as follows:

```
        1) /C=de/O=siemens/OU=ap11/CN=schmid
Surname             :  Henry Schmid
Telephone -Number   :  122345
:  789378
```

The following example shows the output of a **search** command that searches the subtree starting from the object **/C=de/O=siemens/**

OU=ap11 for all attributes with the *Common-Name* **Brown** or **Andrews**. The example also demonstrates the formatting produced by **-pretty** option:

```
% search /C=de/O=siemens/OU=ap11 -pretty -subtree \
–filter {{CN=Brown || CN=Andrews}} -allattr
```

```
 1) /C=de/O=sni/OU=ap11/CN=Brown
Object-Class              : MHS-Distribution-List
: Top
Common-Name               : Brown
MHS-DL-Submit-Permission
Permission-Type         : 0
      Individual
Directory-Name        : /C=de/O=sni
ADMD-Name             : dbp
Common-Name           : Alfred Brown
Country-Name          : de
Domain-Type-1         : MS MAIL
Domain-Value-1        : Brown
Generation            : 396
Given-Name            : Alfred
Initials              : P.
Organization          : Siemens Nixdorf
Organizational-Unit-1 : Munich
Organizational-Unit-2 : P1
Organizational-Unit-3 : P4
Organizational-Unit-4 : ap113
PRMD-Name             : sni
Surname               : Brown
MHS-OR-Address
ADMD-Name               : admd
Country-Name            : de
PRMD-Name               : prmd
Description               : Software Consultant
2) /C=de/O=sni/OU=ap11/CN=Andrews
Object-Class              : MHS-Distribution-List
```

```
: Top
Common-Name               : Andrews
MHS-DL-Submit-Permission
Permission-Type         : 0
Individual
Directory-Name        : /C=de/O=sni
ADMD-Name             : dbp
Common-Name           : Peter Andrews
Country-Name          : de
Domain-Type-1         : MS MAIL
Domain-Value-1        : Peter
Generation            : 396
Given-Name            : Peter
Initials              : P.
Organization          : Siemens Nixdorf
Permission-Type         : 0
Individual
Directory-Name        : /C=de/O=sni
ADMD-Name             : dbp
Common-Name           : Peter Andrews
Country-Name          : de
Domain-Type-1         : MS MAIL
Domain-Value-1        : Peter
Generation            : 396
Given-Name            : Peter
Initials              : P.
Organization          : Siemens Nixdorf
Organizational-Unit-1 : Munich
Organizational-Unit-2 : P1
Organizational-Unit-3 : P4
Organizational-Unit-4 : ap113
PRMD-Name             : sni
Surname               : Andrews
MHS-OR-Address
ADMD-Name               : admd
Country-Name            : de
PRMD-Name               : prmd
Description               : Project Leader
```

show The following example reads the current object and displays the values for the **SN** and **BC** attributes:

```
show -attribute SN BC
```

The example produces the following TCL list:

```
{/C=de/O=sni/OU=ap11/CN=schmid SN=schmid {BC=Software \
Consultancy}}
```

The **-pretty** option produces the following output for the previous example:

```
        1)  /C=de/O=sni/OU=ap11/CN=schmid
Surname           : schmid
Business-Category : Software Consultancy
```

Return Values

Each operation on the **x500obj** object returns one of the following:

- A list of information requested by the user (such as the results of a **search** operation)
- NULL (indicating successful completion of an operation)
- An error message string

The **gdscp** command uses the TCL native error handling facility to log additional error information. This additional information is stored in the two variables: *errorInfo* and *errorCode*. The *errorInfo* variable contains a stack trace of each of the nested calls to the TCL interpreter when the error occurred. The *errorCode* variable is a TCL list containing two elements: **GDSCP** (identifying the **gdscp** program) and the numeric value of the error code. You can use the TCL **catch** command to determine the successful completion or failure of the various **gdscp** commands. Refer to **gdscp.h** header file for a description of the error codes.

Use the **-pretty** option to view the results in a structured format (for example, after a successful **search** operation). The output of a command result in structured format is output in pages of 23 lines in length.

If you specify the **-pretty** option, the return value of the command will be **NULL and not a TCL list.**

You can scan through the output by using special **gdscp** scrolling commands:

n View the *n* th page

-n Skip *n* pages backward

+n Skip *n* pages forward

$ View the last page

q Quit

<Space Key>
Advance to the next page

<CR> Advance one line

Related Information

Refer to the *DCE 1.2.2 Administration Guide—Core Components* for information about the basic concepts and features of TCL.

Refer to the **gdscp**, the **x500abbr**, and the **x500svc** reference pages.

x500svc

Purpose

represents the Service Controls that are passed in a directory operation

Synopsis

x500svc help [*operation-name*] [**-verbose**]

x500svc modify *-option value*

x500svc operations

x500svc show [**-pretty**]

Operations

help Displays the help text for the **x500svc** object and its operations.

If help is invoked without any argument or an option, then it returns a one-line-per-operation help message as follows:

```
modify          Modifies the service control settings.
show            Displays the service control settings.
help            Displays help text for the x500svc
object and its operations.
operations      Lists the operations that can be
performed on x500svc object.
```

If help is invoked with an operation name as the argument, it returns a one line per option help message.

For example:

```
x500svc help show
-pretty     Displays the result in a structured format.
```

Finally, if help is invoked with the **-verbose** option, then a one paragraph description of the **x500svc** object is returned. This command will return text explaining what the object represents and how to use it.

modify Modifies the service controls.

operations Displays a list of operations that can be performed on the **x500svc** object type.

show Displays the service control settings.

If you specify the **-pretty** option, the output is formatted.

If you do not specify the **-pretty** option, the output is presented as a TCL list. Each item in the returned TCL list will consist of the service control and its value.

For example :

```
{{automaticcontinuation TRUE}  {cacheclass NORMAL} ...}
```

Arguments

operation-name
Specifies the operation about which help text is displayed.

Options

You can modify service controls by using the following options and value arguments:

-automaticcontinuation
Processes continuation referrals automatically. Valid values are TRUE or FALSE. The default value is TRUE.

-cacheclass
Specifies the storage class of the DUA cache. Valid values are **RESIDENT**, **PRIVILEGE**, **NORMAL**, and **NONE**. The default value is **NONE**.

-chainingprohibited
Prohibits the use of chaining. Valid values are TRUE or FALSE. The default value is TRUE.

-default Sets all the service control settings to the default values.

-dontdereferencealias
Does not dereference aliases found in the path of a query. Valid values are TRUE or FALSE. The default value is FALSE.

-dontusecopy
Prohibits the use of the shadow entry of the object. Valid values are TRUE or FALSE. The default value is TRUE.

-duacache Uses the DUA cache. Valid values are TRUE or FALSE. The default value is FALSE.

-duafirst Uses the DUA cache first. Valid values are TRUE or FALSE. The default value is FALSE.

-localscope Limits the operation to the bind DSA. Valid values are TRUE or FALSE. The default value is FALSE.

-preferadmfunctions
Prefers administration functions. Valid values are TRUE or FALSE. The default value is FALSE.

-preferchain
Directs the DSA to chain the operation if required. Valid values are TRUE or FALSE. The default value is FALSE.

-priority Specifies the priority of a request. Valid values are **LOW**, **MEDIUM**, or **HIGH**. The default value is **MEDIUM**.

-scopeofreferral
Specifies the scope of referrals. Valid values are **COUNTRY**, **DMD**, or **UNLIMITED**. The default is **UNLIMITED**.

-sizelimit Sets the size limit to the specified value. If you set the value to **INFINITE**, it indicates no size limit. The default is **INFINITE**.

-timelimit Sets the time limit of the specified value. If you set the value to **INFINITE**, it indicates no time limit. The default is **INFINITE**.

-usedsa Uses the DSA. Valid values are TRUE or FALSE. The default value is TRUE.

Refer to the *DCE 1.2.2 Application Development Guide* for a detailed explanation of the service controls.

The **x500svc** object also supports the following options:

-pretty Displays results in a structured format. This option should only be used with the **show** operation.

-verbose Displays a description of the **x500svc** object when used with the **help** command as follows:

```
x500svc help -verbose
```

Description

The **x500svc** object represents the service controls that are passed in a directory operation (such as **search**, **list**, and so on). Whenever you perform any Directory operation, you specify the service controls that are associated with the operation. Service controls determine if an operation is performed on the DUA cache or the DSA, if referrals are processed automatically, the time and size limit of the operation, and so on.

You can use the **x500svc modify** command to change one or more of the service control settings so that all subsequent Directory operations use the new settings. You can also display the current values of the service controls by using the **x500svc show** command.

Examples

1. The following example displays the help text for the **show** operation on the **x500svc** object:

```
x500svc help show
-pretty     Displays the result in a structured format.
```

2. The following example displays the service control settings:

x500svc show -pretty

```
Automatic Continuation   - TRUE
Cache Class    - NORMAL
Chaining Prohibited   - TRUE
Don't Dereference Alias   - FALSE
Don't Use Copy    - TRUE
Use DUA Cache    - FALSE
Read DUA Cache First   - FALSE
Local Scope    - FALSE
Prefer Administration Functions  - FALSE
Prefer Chaining    - FALSE
Priority of Request   - MEDIUM
Scope of Referral   - NONE
Size Limit    - INFINITE
Time Limit    - INFINITE
Use DSA     - TRUE
```

3. The following example changes the default settings so that the DSA is advised to chain the operation if required. The time limit for the completion of any operation is 100 seconds. The size limit of the result is 10 entries.

x500svc modify -preferchain TRUE -timelimit 100 -sizelimit 10

Return Values

All operations on the **x500svc** object return one of the following:

- A list of information requested by the user
- NULL (indicating successful completion of an operation)
- An error message string

The **gdscp** command uses the TCL native error handling facility to log additional error information. This additional information is stored in the two variables: *errorInfo* and *errorCode*. The *errorInfo* variable contains a stack trace of each of the nested calls to the TCL interpreter when the error occurred. The *errorCode* variable is a TCL list containing two elements: **GDSCP** (identifying the **gdscp** program) and the numeric value of the error code. You can use the TCL **catch** command to determine the successful completion or failure of the various **gdscp** commands. Refer to **gdscp.h** header file for a description of the error codes.

Use the **-pretty** option to view the results in a structured format. If you specify the **-pretty** option, the return value of the command will be **NULL** and not a TCL list.

Related Information

Refer to the *DCE 1.2.2 Administration Guide—Core Components* for information about the basic concepts and features of TCL.

Refer to the **gdscp**, the **x500obj**, and the **x500abbr** reference pages.

Index

A

C

D

F

G

I

K

L

M

O

P

R

S

T

U

W

X