NIST Special Publication 500–228

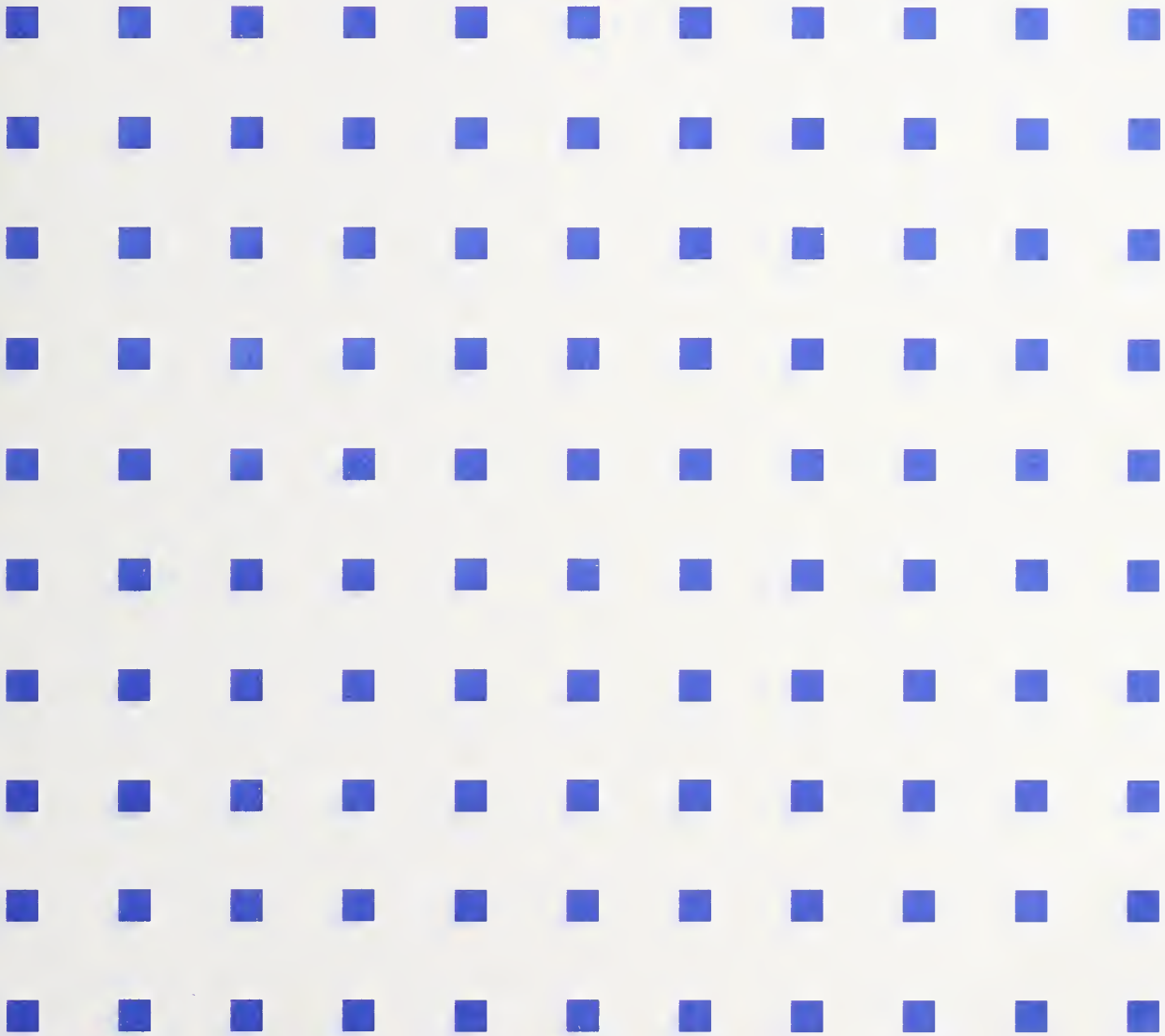
Computer Systems Technology

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology

NIST

Guidelines for the Evaluation of X.500 Directory Products

John Tebbutt



The National Institute of Standards and Technology was established in 1988 by Congress to "assist industry in the development of technology . . . needed to improve product quality, to modernize manufacturing processes, to ensure product reliability . . . and to facilitate rapid commercialization . . . of products based on new scientific discoveries."

NIST, originally founded as the National Bureau of Standards in 1901, works to strengthen U.S. industry's competitiveness; advance science and engineering; and improve public health, safety, and the environment. One of the agency's basic functions is to develop, maintain, and retain custody of the national standards of measurement, and provide the means and methods for comparing standards used in science, engineering, manufacturing, commerce, industry, and education with the standards adopted or recognized by the Federal Government.

As an agency of the U.S. Commerce Department's Technology Administration, NIST conducts basic and applied research in the physical sciences and engineering, and develops measurement techniques, test methods, standards, and related services. The Institute does generic and precompetitive work on new and advanced technologies. NIST's research facilities are located at Gaithersburg, MD 20899, and at Boulder, CO 80303. Major technical operating units and their principal activities are listed below. For more information contact the Public Inquiries Desk, 301-975-3058.

Office of the Director
- Advanced Technology Program
- Quality Programs
- International and Academic Affairs

Technology Services
- Manufacturing Extension Partnership
- Standards Services
- Technology Commercialization
- Measurement Services
- Technology Evaluation and Assessment
- Information Services

Materials Science and Engineering Laboratory
- Intelligent Processing of Materials
- Ceramics
- Materials Reliability[1]
- Polymers
- Metallurgy
- Reactor Radiation

Chemical Science and Technology Laboratory
- Biotechnology
- Chemical Kinetics and Thermodynamics
- Analytical Chemical Research
- Process Measurements[2]
- Surface and Microanalysis Science
- Thermophysics[2]

Physics Laboratory
- Electron and Optical Physics
- Atomic Physics
- Molecular Physics
- Radiometric Physics
- Quantum Metrology
- Ionizing Radiation
- Time and Frequency[1]
- Quantum Physics[1]

Manufacturing Engineering Laboratory
- Precision Engineering
- Automated Production Technology
- Intelligent Systems
- Manufacturing Systems Integration
- Fabrication Technology

Electronics and Electrical Engineering Laboratory
- Microelectronics
- Law Enforcement Standards
- Electricity
- Semiconductor Electronics
- Electromagnetic Fields[1]
- Electromagnetic Technology[1]
- Optoelectronics[1]

Building and Fire Research Laboratory
- Structures
- Building Materials
- Building Environment
- Fire Safety
- Fire Science

Computer Systems Laboratory
- Office of Enterprise Integration
- Information Systems Engineering
- Systems and Software Technology
- Computer Security
- Systems and Network Architecture
- Advanced Systems

Computing and Applied Mathematics Laboratory
- Applied and Computational Mathematics[2]
- Statistical Engineering[2]
- Scientific Computing Environments[2]
- Computer Services
- Computer Systems and Communications[2]
- Information Systems

[1] At Boulder, CO 80303.
[2] Some elements at Boulder, CO 80303.

NIST Special Publication 500–228

Guidelines for the Evaluation of X.500 Directory Products

John Tebbutt

Systems and Network Architecture Division
Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

Funding for this document was provided by
the Department of Agriculture

May 1995

U.S. Department of Commerce
Ronald H. Brown, Secretary

Technology Administration
Mary L. Good, Under Secretary for Technology

National Institute of Standards and Technology
Arati Prabhakar, Director

Reports on Computer Systems Technology

The National Institute of Standards and Technology (NIST) has a unique responsibility for computer systems technology within the Federal government. NIST's Computer Systems Laboratory (CSL) develops standards and guidelines, provides technical assistance, and conducts research for computers and related telecommunications systems to achieve more effective utilization of Federal information technology resources. CSL's responsibilities include development of technical, management, physical, and administrative standards and guidelines for the cost-effective security and privacy of sensitive unclassified information processed in Federal computers. CSL assists agencies in developing security plans and in improving computer security awareness training. This Special Publication 500 series reports CSL research and guidelines to Federal agencies as well as to organizations in industry, government, and academia.

National Institute of Standards and Technology Special Publication 500-228
Natl. Inst. Stand. Technol. Spec. Publ. 500-228, 69 pages (May 1995)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1995

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

Contents

List of Figures

Chapter 1

Introduction

This document provides readers with the means to evaluate various X.500 products and make informed choices as to which available products best match the requirements of their organizations.

This document is aimed at procurement officials, systems administration staff and others involved in the process of obtaining or recommending X.500 products for use in their organizations.

At the time of writing, the 1994 Edition of the X.500/Directory standard, Reference [1], has been finalized, and a number of 1994 conformant products should be arriving in the marketplace within the foreseeable future (though few if any are currently available).

At the same time, the requirement for a standardized, intelligent, distributed database system is increasingly being felt in organizations large and small. The initial requirement is usually for a name-to-address mapping tool for X.400 electronic mail, but as new applications continue to be deployed, such as Electronic Data Interchange (EDI), the Directory is rapidly becoming a pivotal information storage and retrieval medium.

Given that an organization has identified the need for some X.500/Directory functionality, in the form of clients and/or servers, the question becomes one of how best to select a product (or combination of products) which offers an array of features which best match the organization's requirements.

This document has the following structure. In Chapter 2 a brief tutorial on the X.500/Directory standard is presented, the purpose of which is to introduce the various terms and concepts associated with the Directory. Directory entries and their structure are described, followed by the hierarchical organization of the Directory information, and the rules governing this organization (the Directory Schema); the distributed operation of the Directory is outlined; finally, security models and the replication of Directory information are described.

Chapter 3 is entitled "Functional Evaluation of X.500"and contains two broad sections. Section 3.1 details a list of standard Directory functions that a purchaser should require of any Directory product, since they are mandated in the Directory standard. Such functions range from the obvious (e.g., does the product enable the user to perform a read operation) to the less obvious (e.g., does the product enable the user to specify that replicated information is unsatisfactory in response to a given operation). In short, this section deals with the product's *conformance* to the X.500 standard (Reference [1]) and related documents, such as the Industry/Government Open Systems

Specification (IGOSS) (Reference [11]) and the Stable Implementation Agreements from the Open Systems Environment Implementors' Workshop (OIW) (Reference [12]).

A product may conform to the letter of a standard and any associated implementor agreements, Application Programming Interfaces (APIs), etc., yet still be cumbersome to use, time consuming to master, and resource-intensive in maintenance and administration, to name just a few potential pitfalls. Section 3.2 suggests a list of features which, while not mandated in any standard, are desirable in a Directory product simply because they increase the ease of use, maintenance, etc., of the product, and thus increase its overall *usefulness.*

Chapter 4 details how to apply the information set out in the Functional Evaluation Chapter in a practical assessment of an X.500 product, and contains some suggestions as to how to decide on which aspects of the X.500/Directory system are important to the organization in question. In Chapter 4 a measure methodology is developed which enables a meaningful comparison between products, and thus permits the user to arrive at an informed decision as to which product to select. This section is entitled "Performance Evaluation of X.500 Products."

1.1 Acknowledgments

The author wishes to thank Carol A. Edgar of the National Institute of Standards and Technology who assisted in the editing and the preparation of the text for publication. Miss Edgar's unfailing support in both the technical and editorial review of this document was extremely helpful.

Chapter 2

An Introduction to X.500(1994)

2.1 Overview

2.1.1 What is the X.500 Directory ?

In essence the Directory[1] is a distributed database, capable of storing information about people and objects in various nodes or servers distributed across a network. It is these servers, acting in concert, which provide the potentially global access to information made possible by X.500 technology (see fig. 2.1).

Distributing information in this manner has various advantages over the conventional method of centralizing information storage, for example:

- The information is kept "close" to those people or processes which are most likely to make most frequent use of it and are most likely to be responsible for keeping it up-to-date – this is likely to reduce access time and network costs, and increase the likelihood of the accuracy of the stored information;
- Since the information is distributed across several (possibly hundreds, thousands, or even more) servers, the impact of a given server becoming inactive, for whatever reason, is only to make unavailable the information for which that server is responsible, rather than bringing down the entire database, as would be the case if a centralized server were to go down;
- The Directory has the capacity to grow indefinitely in size and storage capacity through the simple addition of new nodes. Such growth might be achievable but would be less practical with a centralized system;
- The Directory offers the opportunity to unify information resources across the globe, as opposed to the insularity which tends to occur when organizations rely on proprietary, centralized databases.

The physical location of the accessed information is transparent to the user. The Directory possesses the necessary knowledge to locate requested information, regardless of where the information might

[1] Throughout this text the terms "X.500" and "The Directory" will be used synonymously.

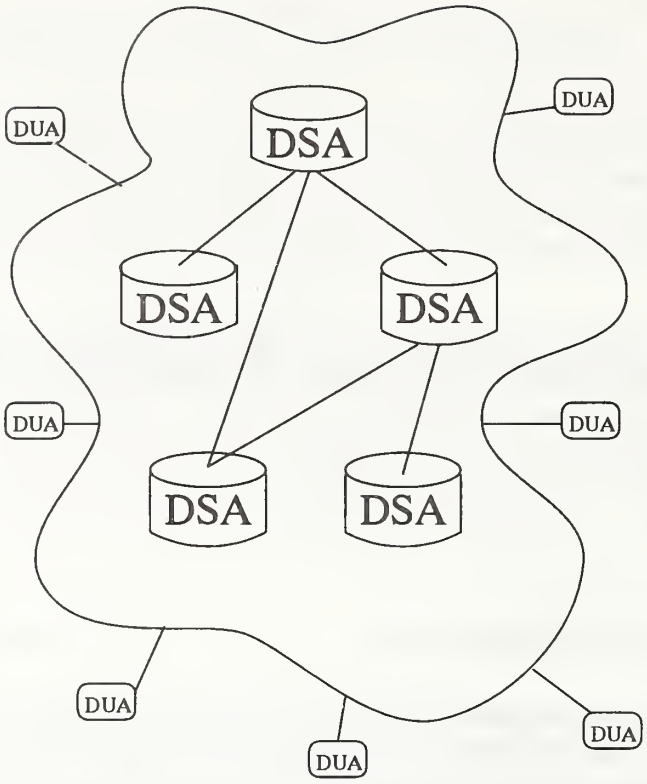


Figure 2.1: *The Directory – Conceptual View.*

be on the network. To access the Directory, the user will usually interface to the server closest to him/her, and all information received will usually appear as if it came directly from that server. From the user's point of view, the Directory behaves exactly as if the user was accessing a centralized server.[2]

2.1.2 How does the Directory work?

The Directory user (person or process) accesses the Directory via a client process known as a Directory User Agent, or DUA (see fig. 2.2). The DUA interfaces with the Directory using a standard protocol between itself and one of the Directory servers, termed Directory System Agents, or DSAs. Usually the DSA contacted would be the one closest, in terms of connection cost or organizational affiliation, to the Directory User Agent.

The DSAs making up the Directory also communicate via a standard protocol which embodies a set of operations which may be performed by the Directory (e.g., retrieving information, adding information, etc.). Each DSA knows how to contact one or more additional DSAs (at least one). This is the mechanism through which a Directory request can be propagated through the system: if a particular DSA is unable to satisfy a request, the request is forwarded to another DSA which is more likely to have the necessary information, and so on.

[2]It is envisioned that most Directory users will not be people, but application processes. Nevertheless, the uniform interface provided by the Directory is still desirable

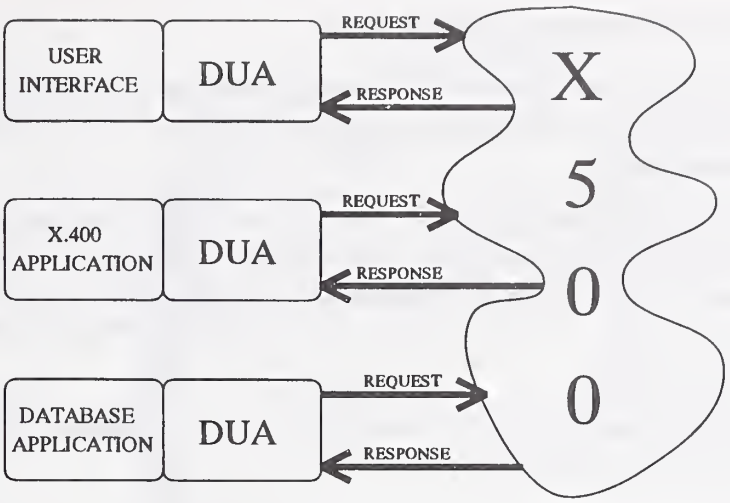


Figure 2.2: *Various applications interface to the Directory via Directory User Agents.*

2.1.3 About this tutorial

The tutorial provides an introduction to the following aspects of X.500:

- **Entries.** The universal unit of information storage in the Directory.
- **Attributes.** The constituent elements of entries.
- **Object Classes.** An identifier signifying the class of entries to which a given entry belongs. Entries are grouped into classes on the basis of shared properties.
- **Schema.** The set of rules and assertions governing where an entry may be placed within the Directory, how it shall be named, and what information it may contain.
- **Services.** The set of services which enable the user to manipulate information held by the Directory.
- **Distributed Operation.** The mechanisms by which multiple separate servers are linked together to form a single Directory entity.
- **Security.** The methods by which the Directory protects the information it holds from unauthorized access.
- **Replication.** The means by which information held by a DSA may be copied to one or more other DSAs in order to increase efficiency and decrease access time.
- **Operational Bindings.** Bilateral agreements made between DSAs to provide various services to one another. These agreements are usually used for the creation and administration of Shadowing agreements.

2.1.4 Tutorial Objective and Constraints

1. The aim of this tutorial is to give a good working knowledge of the principles of the Directory – there is no emphasis on technical details, though references to the Standard document are given when appropriate.

2. It is impossible, in a brief tutorial such as this, to cover in detail all the material and concepts presented in the X.500(1994) Standard. The Standard document itself approaches the thickness of a telephone book! For greater detail, it is suggested the reader consult the Standard itself, Reference [1]. Also, there are a variety of commercially available texts which cover the material in greater depth.

3. This tutorial will be published **both** separately as a self-contained tutorial **and** as part of NIST's *X.500 Evaluation Guidelines* publication.

2.2 Entries

The *entry* is the fundamental unit of information in the Directory. That is to say, all information stored in the Directory is stored in the form of entries, each of which is uniquely named and represents one of three things:

1. An object in the real world – the bulk of the entries in the Directory are of this type, termed an *object entry*;

2. An alternative name for an object entry. This type of entry is termed an *alias entry*, and is also uniquely named; or

3. A collection of information used to meet administrative needs of the Directory. Such an entry is termed a *subentry*.

The unique naming of entries is described in Section 2.5.

2.2.1 Object Entries

Typically, when the term "entry" or "Directory entry" is used, it refers to an object entry, though all entries are structured in the same manner.

All Directory entries are made up of a collection of *attributes* (described in sec. 2.3), each of which describes a particular quality or aspect of the object represented by the entry. Figure 2.3 displays a schematic of an object entry together with some of its constituent attributes and their associated values. The figure represents a hypothetical object entry which has an attribute of type *Telephone Number* with a value of *303-498 1633*, indicating that the real world person in question has the telephone number 303-498 1633. In this regard, an entry may be likened to a database record, and an attribute to a field of that record.

Initially it was envisioned that the entries in the Directory would contain information relating largely to data communications devices and applications, but increasingly it is being thought of as

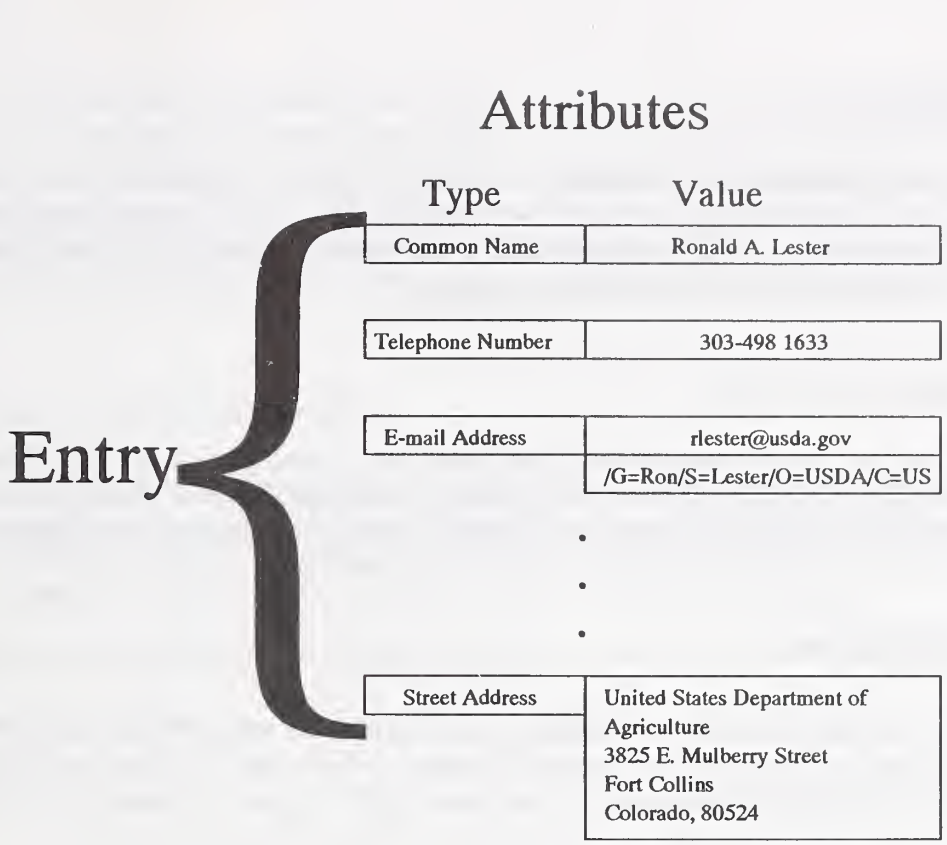


Figure 2.3: *The structure of a Directory entry.*

a general purpose repository for all kinds of information, ranging from personnel records through automotive parts inventories to on-line telephone directories and database services. That having been said, it seems that the most important role for X.500 in the forseeable future is as an enabling technology for data communications, especially electronic mail.

2.2.2 Alias Entries

An alias entry contains only one user attribute, the value of which is the unique name of an object entry. Alias entries are thus used to provide alternative names for object entries. This can be useful in the case when the actual name of an entry is cumbersomely long. The use of aliases will become clearer in Section 2.5 on the Directory Schema.

When an alias entry is encountered by one of the Directory services (decribed in sec. 2.6) as the name of an entry, the *aliasedEntryName* attribute is read, the target object of the operation is reset to the name it contains, and processing continues as if the aliasedEntryName had been supplied as the original entry name to the service. This process is known as *alias dereferencing.*

Aliases may point to object entries or other alias entries, but they must always be leaf entries, i.e., they are not permitted to have subordinate entries.

2.2.3 Subentries

A subentry is used for internal administration within the Directory architecture. It contains information used by the Directory in various administrative functions. For example, *collective attributes*, described in Section 2.3, are stored in subentries, along with accounting and access control information for the entries in the vicinity of the subentry.

2.2.4 Entry Collections

An *entry collection* may be formed when any set of entries share properties or characteristics in common. Usually when this is the case, the common information may be stored once in a *collective attribute* (see sec. 2.3) rather than being stored many times, once in each individual entry.

2.3 Attributes

The Directory Standard specifies and defines a set of commonly used attribute types which can be augmented by various implementor bodies, by vendors or by users. The mechanism for attribute definition is the `ATTRIBUTE` information object class, which is given in Clause 12.4.6 of Reference [3].

The Directory attribute holds information regarding a particular quality or characteristic of an object represented by a Directory entry, and has two principal components: a *type* indicator, and a set of *values*. Figure 2.3 in Section 2.2 shows how attribute types and values are combined into a Directory entry. Also associated with each attribute are a set of *matching rules*, which indicate whether the attribute is to have a single or multiple values. The attribute may also contain administrative data.

2.3.1 Attribute Type

The attribute type is a unique identifier which has both a semantic and a syntactic function. The semantic function is to indicate what purpose the information contained in the attribute serves. For example, is it a telephone number, a network address, a social security number, a nickname, and so on. The syntactic function is to indicate how the attribute value should be parsed in order to accurately retrieve the information it contains.

2.3.2 Attribute Values

The attribute value is where the attribute's information is held. As mentioned above, the parsing and interpretation of the attribute value is determined by the attribute type.

2.3.3 Matching Rules

Each attribute has associated with it a set of matching rules which determine how values of the attribute may be matched against other values. For example, an entry may be of interest if it has

an attribute representing a telephone number, one of whose values is equal to *301-555 8937*. In this case, it would be desirable to permit the comparison of telephone number attribute values for equality with presented values. Such a comparison is dictated by the *equality matching rule* for the attribute in question. The other matching rules which may be included in the attribute are rules for *ordering match*, to determine whether a stored quantity is greater or less than a presented quantity; and *substrings match*, which enables presented substrings to be picked out of stored string values.

2.3.4 Attribute Type Hierarchies

When defining attributes in the Directory, it is possible, though not necessary, to create an *attribute hierarchy*, by defining groups of attributes of a general nature and then defining *subtypes* of these attributes to hold more refined or detailed information of a similar nature. These latter attribute types can, in turn, serve as *super types* for another level of attribute sub-typing.

As an example, suppose we define an extremely general attribute type, which we call *Address* (see fig. 2.4).

Figure 2.4: *The base attribute of the* Address *attribute hierarchy.*

Address can mean virtually any point of contact with some person or entity, and we reflect this by creating several subtypes of our base attribute: *Postal Address*, *Electronic Mail Address*, and *Telephone Number*. Thus we have a two tier hierarchy, as shown in Figure 2.5.

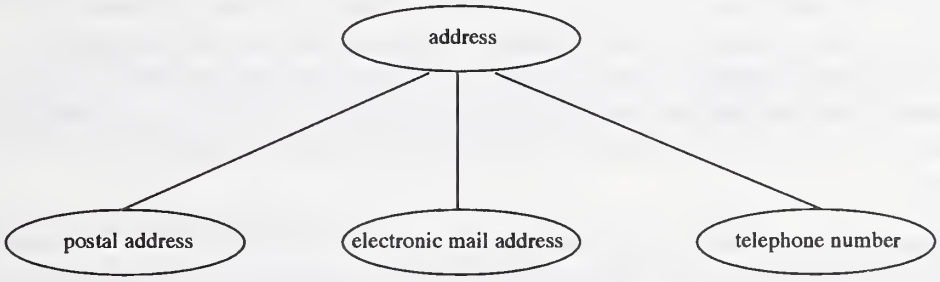


Figure 2.5: *The* Address *attribute hierarchy, showing three subtypes of the base type.*

Of course, each of the three subtypes is also fairly general, for example, we might wish to define individual attribute subtypes under Electronic Mail Address for *X.400 Address*, *Simple Mail Transport Protocol (SMTP) Address* and *cc:Mail Address*, which leads to the three tier hierarchy shown in Figure 2.6.

The useful aspect of attribute type hierarchies is that they offer the opportunity to examine Directory information at different levels of detail. When an attribute type is requested in an information retrieval request,[3] attributes of that type *together with all its subtypes* are returned. Thus, in our example, if the attribute type *Address* is requested, the Directory will return all attributes of types

[3]Information retrieval services are discussed in Section 2.6

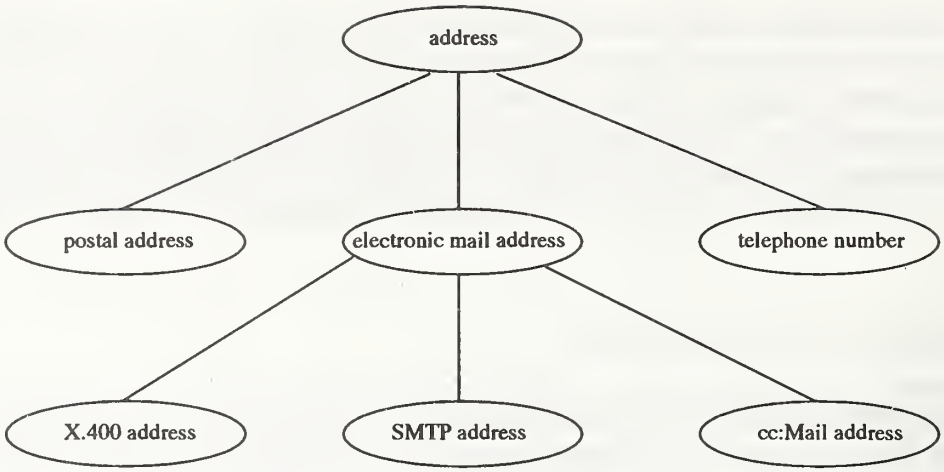


Figure 2.6: *The* Address *attribute hierarchy, showing subtypes of* electronic mail address.

Address, Postal Address, Electronic Mail Address, Telephone Number, X.400 Address, SMTP Address and *cc:Mail Address*. If *Electronic Mail Address* is requested, all attributes of types *Electronic Mail Address, X.400 Address, SMTP Address* and *cc:Mail Address* will be returned. And finally, if only attributes of type *X.400 Address* are desired, that is the attribute type which should be specified in the information retrieval request.

2.3.5 Collective Attributes

Collective attributes serve as repositories for information common to all entries in an entry collection (see sec. 2.2). For example, if all entries in a group share the same telephone number (as may be the case for the members of a household), the information might be stored in a collective attribute residing in a Directory subentry close to the object entries making up the entry collection. One advantage of storing information in this way is that, if it becomes necessary to change the information, this need be done only once for all the members of the collective, rather than once for *each* member.

The information held in collective attributes appears as if it is part of the information held in any entry in the entry collection when that entry is retrieved using any of the Directory interrogation services, but modification must take place through the subentry where the information is actually held.

2.4 Object Classes

Directory object classes are used principally to distinguish between entries representing different types of objects. Each Directory entry must belong to at least one object class, and contain an attribute, the value(s) of which indicate(s) the object class(es) to which the entry belongs. Following from this descriptive function, the entry's object class serves several specific functions within the Directory:

- It governs the *attributes* the entry contains (see sec. 2.4.1);

- It governs the *position* the entry may take in the Directory structure (see sec. 2.4.2); and
- It governs the *administrative policy* associated with the entry (see sec. 2.4.3).

As is the case for Directory attributes, object classes may be defined in international standards, by other standards or implementor bodies, by vendors, or by users. The mechanism for object class definition is described in Clause 12.3.3 of Reference [3].

2.4.1 Entry Attributes

The object class of an entry dictates which attributes the entry may contain.[4] In fact, it specifies a set of attributes the entry **must** contain, along with a further set which the entry **may** contain. No additional attributes are permitted. See Section 2.4.5 for additional information on object class types and entry composition.

As an example, suppose we define an object class, **Used Car**, to represent a used car. We might conclude that certain information about a vehicle was indispensable in entries of this object class, so we would insist that they **MUST CONTAIN** attributes indicating the car's *Make/Model*, *Year of Manufacture* and *Original Mileage*. On the other hand, there are several attributes the car might have which we consider of lesser importance, but nonetheless desirable, so we stipulate that entries of this object class **MAY CONTAIN** attributes indicating the vehicle's *color*, *engine size*, *list of options*, *availability of service record*, and so on. We have now described the composition of entries used in the Directory to describe used cars.

2.4.2 Entry Position

The object class identifier attribute of the entry is used by the Directory to ensure that the entry is not placed inappropriately within the Directory database. The structure of the Directory database and the placement of entries therein is described in Section 2.5.

2.4.3 Administrative Policy

Various aspects of administrative policy may be associated only with entries of particular object classes. The Directory Administrative Model is described in Clause 4 of Reference [3].

2.4.4 Object Class Inheritance

Object classes may be created by declaring them to be *subclasses* of existing object classes. In this case, the new object class inherits all the attributes of an existing object class, its *superclass*, and has the opportunity to add further attributes.

To extend our example above, suppose we have an organization, Dave's Used Cars, for which the existing **Used Car** object class is useful but not sufficient. A new object class, **Dave's Used Car**, may be derived from the existing one by declaring it a **SUBCLASS OF Used Car**, and further

[4] See also Section 2.5 for the influence of Directory Content Rules on the contents of entries.

stipulating that it **MUST CONTAIN** attributes indicating, say, a local *stock number* and *date of arrival*, and that it **MAY CONTAIN** attributes indicating the vehicle's *source* and perhaps *blue book value*. Thus, entries of object class **Dave's Used Car** will contain all the attributes of the **Used Car** object class, plus those additionally specified by Dave.

Each entry in the Directory must contain an attribute indicating the object classes to which it belongs. This is because, by definition, each object class is a subclass of a special object class known simply as **top**, which indicates that all attributes **MUST CONTAIN** such an attribute. This attribute holds the unique object class identifier for the object class to which the entry belongs, in addition to the corresponding identifiers for all the entry's superclasses. Thus, an entry belonging to a given class also belongs to all the superclasses of that class. This set of superclasses, from the entry up to **top**, is know as the object class' *superclass chain.*

Returning to our example, an entry of class **Dave's Used Car** also belongs to class **Used Car**, and so on, back up the chain of whatever superclasses **Used Car** might have.

One important effect of this object class inheritance is that if an information retrieval request is made based on the object class of the entry, all entries of that class *and all its subclasses* will be returned.

2.4.5 Types of Object Class

Object classes may be subdivided into three types: *abstract, structural*, and *auxiliary.* So far, this section focused on structural object classes. The two other classes can be summarized as follows:

2.4.5.1 Abstract Object Class

An abstract object class is an object class which is defined purely for the purpose of serving as a superclass or template for other (structural) object classes. It is a way of conveniently collecting together a set of attributes which it is known will be common to a set of structural object classes, in order that these classes may be derived as subclasses of the abstract class rather than being defined from scratch. For example, an abstract object class Person can be defined with address, phone number, Date of Birth (DOB), etc., things everyone has. Then it is possible to subclass a structural object class like NISTPerson, with additional attributes of grade, salary, personnel number, etc.

Note that an entry may not belong to an abstract object class.

2.4.5.2 Auxiliary Object Class

Each entry, while belonging to only a single structural object class, may belong to zero or more auxiliary object classes. Auxiliary object classes serve as a means to provide multiple inheritance to Directory entries, that is to say, to combine the attributes from two or more superclass chains into a single entry. This is useful in the case where a common group of attributes is desired in entries from various object class hierarchies. For example, the attributes of an object class **Bank Account Holder** might be included into entries of structural object classes as varied as **Residential Person, Personnel Manager, Business Organization** and **Government Department.**

2.5 The Directory Schema

2.5.1 Overview

The Directory Schema constitutes the framework within which Directory information is stored. It consists of a set of rules and definitions which define the naming of entries, the content of attributes and entries, the structure of the Directory as a whole, and the hierarchical relationships between entries.

The Schema comprises the following components (in accordance with Recommendation X.501 Clause 12.2, Reference [3]):

- **Name Form** definitions, which describe how Directory entries should be named;
- **Directory Information Tree (DIT) Structure Rules**, which define hierarchical relationships between entries of different object classes;
- **Content Rule** definitions, which allow the inclusion in entries of attributes not indicated in the entries' structural object classes;
- **Object Class** definitions – see Section 2.4;
- **Attribute Type** definitions – see Section 2.3; and
- **Matching Rule** definitions – see Section 2.3.

2.5.2 Naming of Directory Entries

2.5.2.1 Directory Names

Each entry in the Directory is identified by at least one[5] unique name, called the entry's **Distinguished Name** or **DN**. The DN is formed in the following fashion:

1. The Directory Information Base (DIB) is organized into a tree-shaped hierarchy, the DIT, in which each entry has exactly one superior entry but may have many subordinate entries. This organization is illustrated in Figure 2.7.

   Clearly, each superior entry may have many subordinates, so the entry may be one of many siblings at the same level in the tree.

2. Each entry contains at least one attribute value which is designated as the entry's name at that level, i.e., relative to all its siblings. This name, known as the entry's **Relative Distinguished Name** or **RDN**, must be unique among all the entry's siblings. There are times when it is necessary to select more than one attribute value in order to distinguish between siblings, and other times when it is desirable to select more than one attribute value for the sake of clarity:

Surname : Lester	OR	**Surname :** Lester, **Given Name :** Ronald

[5] Aliases may be used to provide alternative names.

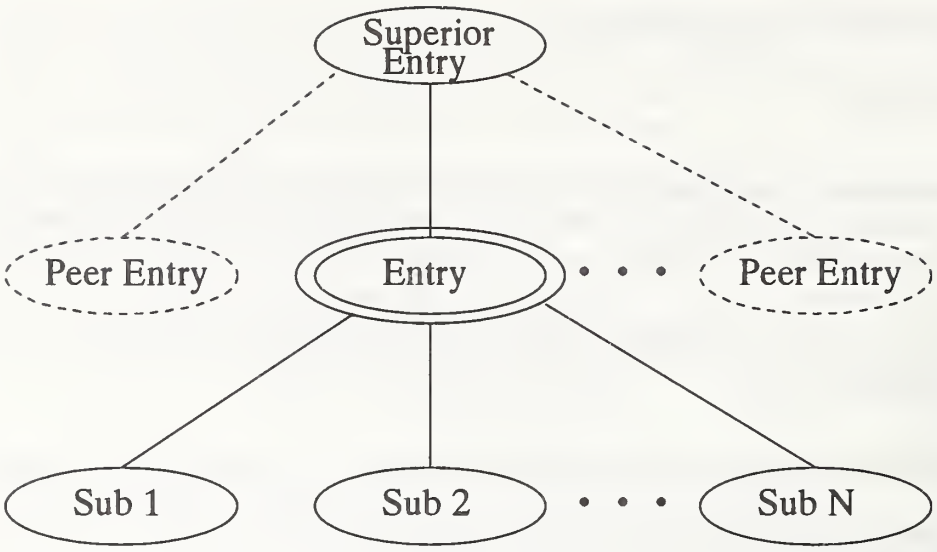


Figure 2.7: *The basic structure of the Directory Information Tree.*

3. The unique name of the entry, its **Distinguished Name** or **DN**, is formed by the concatenation of the RDN of the entry with those of each of its superiors, from the top of the DIT on down to the entry.

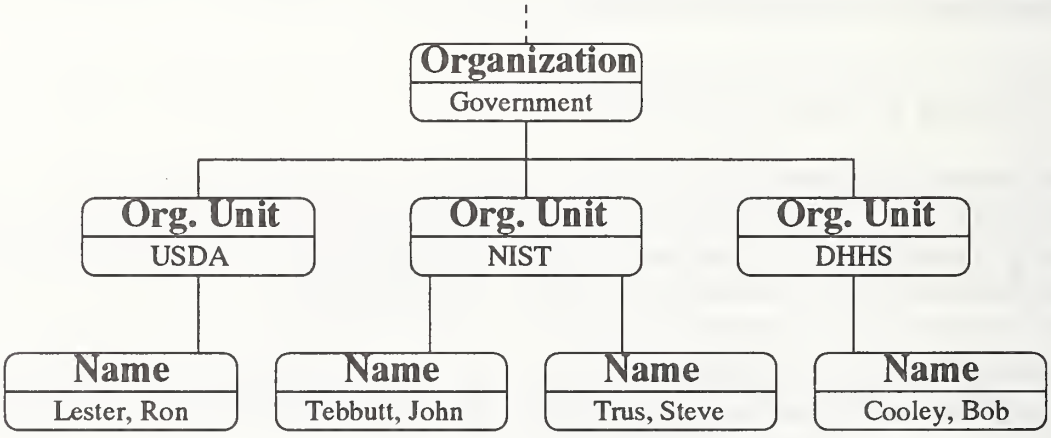


Figure 2.8: *An example of a Directory Information Tree.*

The following example applies these rules to the hypothetical DIT three levels deep shown in Figure 2.8. The example concerns the naming of organizations, organizational units, and people connected with those organizational units.

The example contains eight entries, each with its own Distinguished Name:

1. /Organization (O) = Government
2. /O = Government/Organizational Unit (OU) = USDA
3. /O = Government/OU = NIST

4. /O = Government/OU = DHHS
5. /O = Government/OU = USDA/Name (CN, for "Common Name") = Lester, Ron
6. /O = Government/OU = NIST/CN = Tebbutt, John
7. /O = Government/OU = NIST/CN = Trus, Steve
8. /O = Government/OU = DHHS/CN = Cooley, Bob

This example also illustrates the use of three standard attributes commonly used for the naming of entries: Organization Name (O), Organizational Unit Name (OU) and Common Name (CN). These attributes, together with the other standard attributes (i.e., those defined by the X.500 Standard, specifically Recommendation X.520), can be found in Reference [7].

2.5.2.2 Name Form Definition

A Name Form specifies which attribute types within an object class may form part of the RDN of entries belonging to that class. This carries the implication that there may be certain attributes in an entry which should not be used as part of that entry's RDN. For example, in an entry representing a person's health record, the use of the person's weight or last white blood cell count as part of the RDN would in most cases be of little utility, whereas the use of the person's name, patient number, Social Security number, etc., would be a far more natural and practical choice. The name form definition is used to enforce such constraints within the Directory.

Another implication of the name form definition is that *at least one* of the mandatory attributes of the object class must be specified as the naming attribute for the object class, if entries of that class are to have names (which, of course, they must).

Note also that potentially several name forms may be defined for each object class, allowing for entries belonging to that class to be named in different ways, according to their placement in the DIT and the accompanying DIT structure rules (see sec. 2.5.3).

The formal method of Name Form Specification is given in Clause 12.6.3 of Reference [3].

2.5.3 Directory Information Tree Structure Rules

The placement of entries within a portion of the DIT is governed by rules set out by the responsible authority, known as *DIT Structure Rules*. Each entry in the Directory contains an attribute of Type **governingStructureRule**, which holds the structure rule that governs the possible placement of the entry. The entry may only be placed in a portion of the DIT if the Directory schema holds a DIT structure rule which matches that held in the entry.

The DIT structure rule consists of 3 parts:

1. A unique identifier;
2. A name form identifier, which specifies the name form that entries governed by this structure rule will take;

3. A list of superior structure rule identifiers, which denote where in the DIT the entry may be placed, i.e., the classes of entry to which it is subordinate.

2.5.4 Directory Information Tree Content Rules

The content of an entry, in terms of the attributes it contains, is regulated primarily by the entry's structural and auxiliary object classes (see sec. 2.4). However, additional contents may be specified by the definition of a *DIT Content Rule* associated with the entry's structural object class.

A DIT content rule specifies the following (in accordance with Reference [3], Clause 12.7.1):

- The identifier of the structural object class to which it applies;
- The identifiers of the auxiliary object classes permitted in entries governed by the rule (optional);
- The identifiers of the mandatory attributes required for entries governed by the content rule, in addition to those mandated in the structural and auxiliary object classes (optional);
- The identifiers of the optional attributes required for entries governed by the content rule, in addition to those named in the structural and auxiliary object classes (optional); and
- A list of identifiers of optional attributes from the entry's structural and auxiliary object classes which the content rule *precludes* from appearing in entries governed by the rule (optional).

Note that, unlike the characteristics of an object class, those of a DIT content rule are *not* inherited by any subclasses of the object class to which it applies.

The DIT content rules are thus useful in the following ways:

- They permit the modification of an object class at a particular level in the object class hierarchy, while avoiding the inclusion of the modifications into the object class chain;
- They permit the exclusion of certain optional attributes from entries of a given object class without modifying the object class itself; and
- They allow the inclusion of individual attributes into entries, without reference to other object classes.

2.6 Services

This section describes the set of services available to users of the Directory for the retrieval and manipulation of Directory information. With these services, the user can retrieve information, add new information, browse through the DIB, delete information and move information around within the Directory Information Base.

Note that these services are performed on behalf of the user[6] by the Directory System Agents (DSAs) which administer the Directory Information Base. Access to the Directory is via the Directory User Agents (DUAs), which may present these services in a variety of ways, depending upon the DUA product in use.

Protocols. Directory User Agents and DSAs communicate with each other using standardized communications protocols. The **Directory Access Protocol**, or **DAP**, is used by DUAs to communicate with DSAs, and vice versa. The more complex **Directory Systems Protocol**, or **DSP**, is used by DSAs to communicate with each other. It contains information not found in the **DAP**, such as trace information, which enables DSAs to monitor the progress of an operation.

Application Contexts. An Application Context is a group of functions or operations required in order to provide a particular service or set of services. The Directory standard specifies several Application Contexts. The **directoryAccessAC** Application Context describes the functionality necessary to provide the services associated with the **DAP**, while the **directorySystemAC** Application Context acts in a similar fashion for the **Directory System Protocol**. The remaining Application Contexts are concerned with Shadowing or Replication of data in the Directory, and Directory Operational Binding Management.[7]

The outcome of a Directory operation takes one of three forms:

- **Result.** A Directory result contains either (a) the requested information from the indicated entry or entries, in the case of the **Read**, **List** and **Search** operations, or (b) an indication of whether the operation was successful or not in the case of the remaining operations;
- **Referral.** If the DSA holding the target entry of the operation could not be reached for some reason (e.g., the **chainingProhibited** service control was set), then an indication of the address of the DSA which is logically the next closest to the target entry's DSA is returned to the Directory User Agent;
- **Error.** If the entry could not be located, a potential violation of the schema was detected, a possible security breach was detected, or any of a number of other conditions prohibiting the completion of the operation occurred, this is conveyed to the DUA in the form of a Directory Error.[8]

The Directory offers the following basic services, each of which will subsequently be reviewed in greater detail. It should be emphasized that, in isolation, these services provide only the building blocks from which more sophisticated, value-added user services may be constructed.

- **Read.** Retrieve the information contained in an entry, as specified by its Distinguished Name;
- **Compare.** Compare a user-supplied attribute value against one held in an entry, as specified by its Distinguished Name;

[6] Person or application process.

[7] See Sections 2.10 and 2.9 for more on Shadowing/Replication and Operational Bindings in the Directory.

[8] See Reference [4] for a fuller explanation of Directory Errors.

- **List.** List the subordinate entries of an entry, as specified by its Distinguished Name;
- **Search.** Search through all the subordinate entries of an entry, as specified by its Distinguished Name, returning those entries which match specified criteria;
- **Add Entry.** Add a new entry to the Directory Information Base, specifying the new entry's name and contents;
- **Remove Entry.** Delete an entry from the Directory Information Base, as specified by its Distinguished Name;
- **Modify Entry.** Modify the contents of a Directory entry, as specified by its Distinguished Name, specifying the desired modifications;
- **Modify Distinguished Name.** Change the Relative Distinguished Name of an entry, as specified by its Distinguished name, or move it to a new superior in the Directory Information Tree, or both.

2.6.1 Read

If users know the DN of an entry containing information they wish to retrieve, the *Read* service may be used to obtain the entry's contents or a subset thereof. The user needs to specify the following:

- The DN of the entry;
- The parts of the entry to be retrieved. The user is able to select certain attributes or simply to request the entire contents of the entry. For example, a telephone directory application might obtain John Smith's telephone number from his personnel record entry simply by requesting his telephone number attribute (there is no reason for a telephone directory application to retrieve the entire entry) while a personnel application would need access to the entire entry;[9]
- An optional request to the Directory to return the privileges she/he has to modify the entry and its attributes.[10]

2.6.2 Compare

The *Compare* service is used to compare a value against an attribute value in an entry whose name the user knows. It returns a true or false response – true if the values matched, false if they didn't. Such a service can be used for password checking applications or for other applications where the creator or owner of a Directory entry does not wish to disclose information directly, but is willing to confirm information to individuals who already have access to it. The user specifies:

- The DN of the entry; and
- The value to be compared.

[9] In fact, the telephone application can be actively prevented from retrieving information other than the telephone number attribute using *access control* – this will be discussed in Section 2.8.

[10] This is also tied into access control – see Section 2.8.

2.6.3 List

This service is used primarily to browse the Directory Information Tree. With it, a user can request a listing of the immediate subordinates of an entry, whose name she/he supplies. This process may be repeated using the name of one of the subordinates returned, and so on. The user may thus better pictorialize the part of the DIT in which she/he is interested. This process can also be used as a primitive search function. For example, if a user wants to look up Bob Walsh's project code, and she/he knows Bob works in Accounts, but is not quite sure of Bob's DN, she/he can list out the subordinates of the Accounts Department's entry, and find Bob's entry that way. This service requires the user to specify two items:

- The DN of the entry whose subordinates are to be listed; and
- An optional indication as to whether the results are to be returned in a paged fashion.

2.6.4 Search

The most powerful of the Directory information retrieval services, *Search*, enables the user to extract from a portion of the DIB those entries which conform to a set of criteria she/he has specified, returning selected information from each entry. In essence, the Search operation identifies a set of entries which satisfy the user's requirements, and then reads each entry in much the same way as is done by the *Read* service.

The starting point for the search is an entry whose DN the user supplies. The search criteria are supplied in a standard logical format, using the operators **AND, OR** and **NOT**, and can be grouped. They also depend on the matching rules discussed earlier (see sec. 2.3).[11]

Returning to our earlier example of Dave's Used Car business, suppose that one of Dave's customers is looking for a 1989 Corvette, but will settle for a Camaro, as long as it is a 1990 or newer model and the price is less than $10,000. For the benefit of the Search operation, these criteria are expressed as follows:

((MAKE **EQUAL TO** Corvette) **AND** (YEAR **EQUAL TO** 1989))

OR

((MAKE **EQUAL TO** Camaro)

AND

(YEAR **GREATER THAN** 1989)

AND

(PRICE **LESS THAN** 10,000))

[11] The means by which the user actually communicates the search criteria to the Directory via the DUA will depend upon the product in use, so it is only possible to describe the logical framework here.

Since both cars happen to be Chevrolets in this instance, the DN of the starting entry might look something like:

/C=US/O=Dave's Used Cars/Category=Stock/Manufacturer=Chevrolet[12]

The Search service allows the user to specify which attributes of the matched entries should be returned, just as for the Read service (see sec. 2.6.1) so, to continue our example, Dave might pull up the mileage and stock number of each car returned by the Search.

2.6.5 Add Entry

The *Add Entry* service allows the user to create a new entry in the Directory Information Base. The new entry may only be created as a *leaf entry*, i.e., it must be added as a subordinate of an existing leaf entry, as opposed to being inserted into the DIT between existing entries. Subordinate entries to the new entry may be added later.

In order to create a new entry, the user should specify:

- The DN of the entry;
- The set of attributes which will make up the new entry (see secs. 2.2 and 2.3); and
- The DSA upon which the entry should reside, if this is different from the DSA holding the entry's superior entry (see sec. 2.7 for more information on the distribution of information between Directory System Agents).

The Directory performs a series of checks on the information in the Add Entry request:

- Does the superior entry exist?
- If so, does the user have the authority to create a new entry at this position in the DIT (see sec. 2.8 for information on DIT Security policy)?
- If so, is the new entry allowed as a subordinate to the superior entry, based on their respective object classes?
- If so, are the attributes specified in the Add Entry request consistent with the supplied object class and any applicable DIT Content Rules?
- If so, the entry is added, otherwise the request is rejected.

The Add Entry service is the primary means by which information is added to the Directory once the initial bulk load is done.[13]

[12]For the sake of this example, Category and Manufacturer are Object Classes assumed to have been created by Dave for the classification of the objects in his database.

[13]The means by which the initial bulk load are done will depend on the particular X.500 product in use.

2.6.6 Remove Entry

In order to delete an entry from the DIB, the user employs the *Remove Entry* service of the Directory. As with the Add Entry service, this service may only be applied to leaf entries of the DIT – it is not possible to delete a non-leaf entry, leaving a number of prior subordinates dangling.

All that is required in order to delete an entry is the entry's DN. If the entry exists, is a leaf entry, and the user has sufficient security access rights (see sec. 2.8 for information on DIT Security policy), then the entry will be deleted.

2.6.7 Modify Entry

Information held in the Directory can be updated through use of the *Modify Entry* service. This service may be applied to any entry in the DIB, subject to access control considerations (see sec. 2.8 for information on DIT Security policy), and permits multiple simultaneous modifications to be made. The user must specify the following:

- The DN of the entry to be modified;
- The set of modifications to be made to the entry, which may comprise any number of the following modification types:
  - Add an attribute;
  - Delete an attribute;
  - Add one or more values to an existing attribute;
  - Delete one or more values from an existing attribute.

Note that the replacement of values in an attribute can be achieved by a combination of delete value and add value instructions in the same request. In fact, most user interfaces should hide this entirely from the user and simply offer a *Replace Value?* option.

2.6.8 Modify DN

The *Modify DN* service is used to change the Distinguished Name of a Directory entry. Since the DN of an entry determines its position in the DIT, this service may be used either to change the position of an entry in the DIT, or simply to change the RDN of the entry, leaving its position in the DIT unaffected.

Two simple examples of the use of the Modify DN operation might be a name change:

/O=Accounts

might become

/O=Accounts and Finances

or a structural reorganization:

/O=GMC/Location=Detroit/O=Accounts

might become

/O=GMC/Location=Louisville/O=Accounts.

In order to employ this service, the following information is necessary:

- The original DN of the entry;
- The new RDN to be given to the entry, if the Distinguished Name is being changed;
- Whether the old RDN should be deleted; and
- The DN of the entry's new superior entry, if the entry is to be moved in the Directory Information Tree.

Note that security and schema rule enforcement are carried out by the Directory, so that if the renaming or repositioning of the entry would be in violation of these considerations, the operation will not be carried out (see secs. 2.5 and 2.8).

2.6.9 Common Arguments

Each Directory operation carries with it a parameter set known as **Common Arguments.**[14] The elements making up the **Common Arguments** specify various elements of the operation request which are relatively stable throughout the duration of a Directory interface session. They are referred to as **Common Arguments** because they are common to all operations. Some of the more important elements of the **Common Arguments**, from the user's point of view, are listed here:

- **Service Controls.** The Service Controls element is itself a set of parameters which allow the user to specify, for instance, whether information from a replicated copy of an entry will suffice in lieu of information taken directly from the master copy; whether chaining between DSAs is preferred or prohibited;[15] the priority of the request; the time limit within which completion of the operation is required; the size limit of any operation response, and so on.
- **Security Parameters.** This element contains any security-related information connected with the request.
- **Requestor.** This is the Distinguished Name of the originator of the Directory operation.

[14] See Reference [4], Clause 7.3 for more on Common Arguments.
[15] see Section 2.7 for more on chaining.

2.7 Distributed Operation

So far in this text, mention has been made several times of the fact that the Directory is made up of several Directory Systems Agents (DSAs), each of which holds a portion of the overall Directory Information Base.[16] This section deals with how the DSAs cooperate with one another in order to provide users with transparent access to the data stored in the Directory Information Base.

2.7.1 Relationship of DSAs to the Directory Information Tree

Each DSA represents an *Administrative Authority*, which extends over one or more parts of the Directory Information Tree. The individual parts of the DIT which fall under the authority of a DSA are referred to as *Naming Contexts*.

2.7.1.1 Naming Contexts

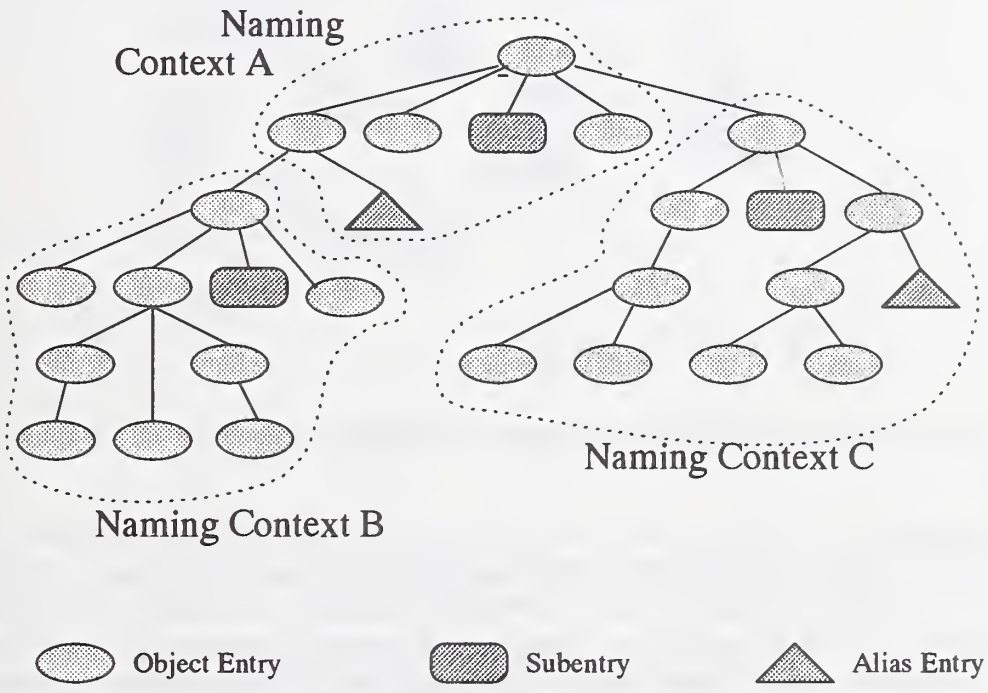


Figure 2.9: *An example DIT showing different Naming Contexts.*

The DIT can be broken up into subtrees,[17] each of which begins with a particular entry as its root and contains all the subordinate entries of that entry, down to a certain level. Such a chunk of the DIT is referred to as a *naming context*, and represents the way the DIT is broken up for storage in the various DSAs comprising the Directory. Figure 2.9 illustrates how a Directory Information Tree may be broken up into various naming contexts. Note that the lower limit of a naming context may be made up of leaf or non-leaf entries. If one of the lowest level entries is a non-leaf, this

[16] There is a degenerate case where all the data are stored in a single DSA, in which case the Directory becomes a centralised, as opposed to a distributed, database, and the information in this section is not relevant.

[17] A subtree is simply a branch of the Directory Information Tree.

implies that it has at least one subordinate entry, which must be in another naming context. Such a naming context is referred to as a *subordinate* naming context as a result, with the previous naming context being referred to as its *superior* naming context. In Figure 2.9, contexts B and C are subordinate to context A, which is their superior. The DN of the root entry of the naming context is called the *Context Prefix*.

2.7.1.2 Relationships Between Directory System Agents

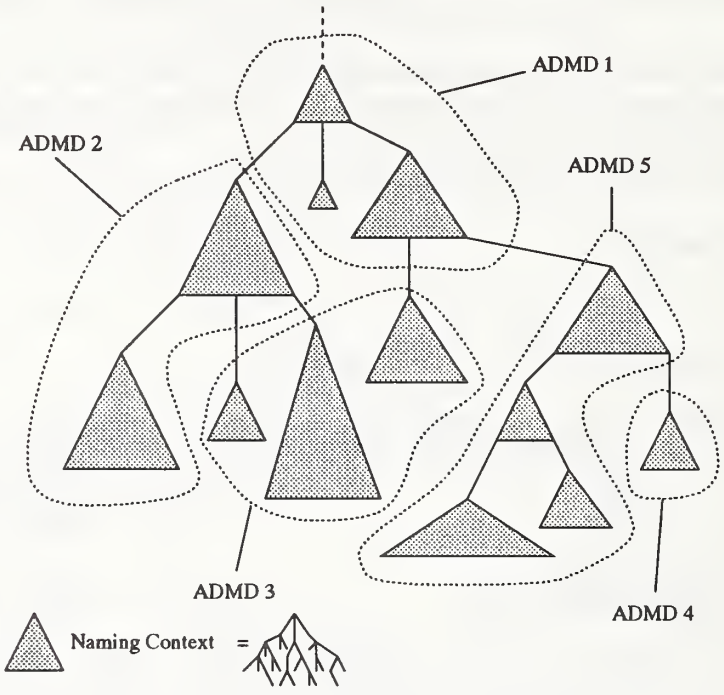


Figure 2.10: *An example DIT showing the mapping of different naming contexts into administrative domains.*

The relationships between DSAs, and thus, between the Administrative Management Domains (ADMDs) they represent are dictated by the relationships between the naming contexts they administer. It should be noted that any given DSA may administer any number of naming contexts from anywhere in the DIT, but the root entry of each naming context will usually have a superior entry in another naming context, which in turn will often reside on another Directory System Agent.[18] In such a situation, and for that particular naming context, the DSA/ADMD holding the superior naming context is said to be the superior DSA/ADMD, while that holding the subordinate naming context is described as the subordinate Directory System Agent/Administrative Management Domain. Figure 2.10 shows an example DIT broken up into five ADMDs (i.e., spread across five DSAs). In this example, it can be seen that ADMD 1 is superior to ADMD 2, ADMD 3 and ADMD 5; ADMD 2 is superior to ADMD 3; and ADMD 5 is superior to ADMD 4. The superior ADMD to ADMD 1 is not shown.

[18]This applies to all naming contexts except the so-called "first level" or "root" naming context, which has the abstract root of the entire DIT as its root. See Clause 17.3 of [3] for further detail.

2.7.2 Knowledge

The DSAs making up the Directory are able to function as a single unit because each one has at least a minimum amount of knowledge about DSAs which hold external naming contexts. The absolute minimum amount of knowledge a DSA must possess is the address of a DSA which holds a naming context which is superior to all the naming contexts held by the DSA, and the addresses of all DSAs which hold naming contexts subordinate to those held by the Directory System Agent. Thus, if a DSA is presented with a request concerning a Directory entry, it knows that it can either process the request itself, because it holds the entry in one of its own naming contexts, or that the entry exists in a naming context subordinate to one it holds, in which case it can forward the request to a DSA responsible for a subordinate naming context or, if it has no idea where the entry might be, as a last resort it can forward the request to a DSA holding a naming context nearer the root of the DIT, in the knowledge that the hierarchical arrangement of the DIT will eventually direct the request to the correct Directory System Agent.

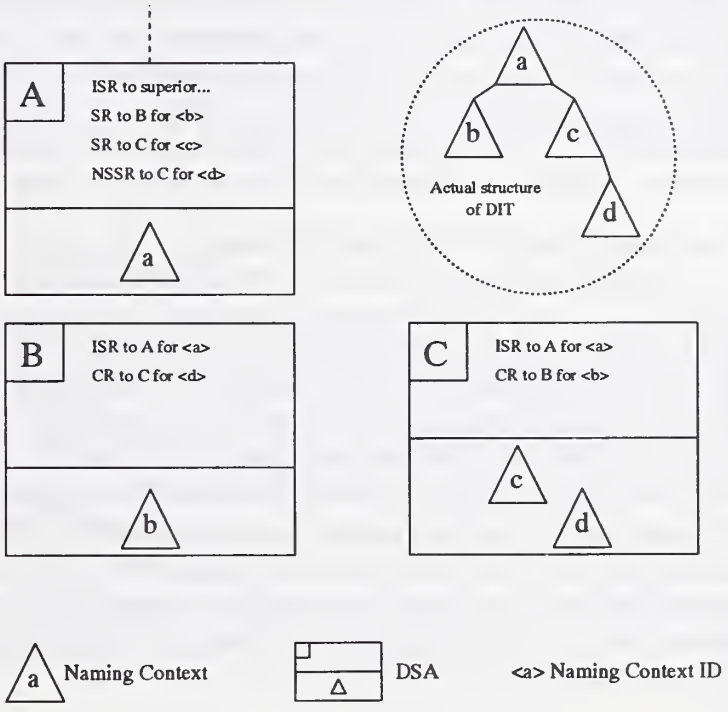


Figure 2.11: *An example DSA configuration showing the knowledge references associated with the DIT structure.*

The types of knowledge described above are known as *Superior (Knowledge) References* and *Subordinate (Knowledge) References*, respectively. Other types of knowledge reference exist, which may be used to enhance the distributed operation of the Directory:

- **Immediate Superior Reference.** This is a knowledge reference to a DSA which holds a naming context immediately superior to one held by the DSA, and can be used when the DSA recognizes that the DN of the target entry of an operation matches part of the context prefix of a naming context it holds, but is shorter;

- **Non-Specific Subordinate Reference.** This is a reference to a DSA which holds a naming context subordinate to one held by the DSA, the context prefix of which is not known by the DSA (in contrast, in a subordinate reference the context prefix of the subordinate naming context *is* known). This type of reference is used in the same way as a subordinate reference;
- **Cross Reference.** This is a reference which may point to a DSA holding a naming context anywhere in the DIT, and is used to optimize the time taken to pinpoint an entry. For instance, if a DSA frequently receives requests pertaining to entries in a particular naming context which is neither subordinate nor superior to a naming context it holds, it my be wise to set up a cross reference to the DSA holding the naming context in question, so that the requests may be forwarded there immediately.

Figure 2.11 shows some knowledge references which might be associated with the mapping of an example DIT onto three Directory System Agents. The DSA A holds naming context A, and thus must hold Immediate Subordinate References to DSAs B and C, since they hold naming contexts B and C, respectively, each of which is immediately subordinate to naming context A. The DSA A also holds an Immediate Superior Reference to an unknown superior (unknown only because it is not shown in the example), and a Non-Specific Subordinate Reference to DSA C for naming context D. Of the references held by DSA A, only the last one, the Non-Specific Subordinate Reference, is optional, and will have been included to increase the efficiency of DIT traversal.

The DSAs B and C each hold Immediate Superior References to DSA A, since DSA A holds the superior naming context to naming contexts B and C, respectively. The DSAs B and C also hold cross references to each other. These Cross References are optional, but they may increase Directory efficiency by enabling each of these DSAs to bypass DSA A and pass information requests directly to one another for entries residing in naming contexts they hold. For example, if DSA C receives a request relating to an entry which lies in naming context B, using the Cross Reference it can pass the request directly to DSA B, whereas the alternative would be to pass the request up to DSA A and rely on its knowledge of naming context B's location in order to complete the request. Clearly this latter alternative would tend to increase resource utilization, error rates and processing time for the request. Note that, since DSA B does not have a Cross Reference to DSA C for naming context D, any requests it receives for entries lying in naming context D will be passed up through DSA A. This emphasizes that a Knowledge Reference of any kind refers to a naming context as well as the DSA which administers it.

2.7.3 Knowledge and Efficiency

The retrieval of Directory information may involve the interconnection of several Directory System Agents. In such a chain of systems, each DSA will do some processing before initiating a connection to the next DSA, and this process can be expected to be time consuming. Just how time consuming it is depends on the quality of the communications links between the systems, the quality of the remote systems themselves, along with the X.500 products in use at the remote sites, and to a large degree on the optimization of the local DSA configuration. With appropriate knowledge references, the number of DSA-DSA hops can be minimized, and the processing time for a distributed operation thus reduced. The inclusion of such knowledge references is a crucial part of the configuration of the local DSA and adds greatly to the system's intelligence.

2.8 Security

Security in the Directory requires two separate components: *authentication* of Directory users to verify their identity, and *access control* procedures to prevent unauthorized access to Directory information. While security is of paramount importance in the Directory, just as in any other open application, discussion of this topic can become lengthy and involved and, since security policy is not a topic unique to the Directory, it will be given only a perfunctory overview here. For an excellent and detailed explanation of authentication practices, the reader is referred to Reference [9], and for a thorough summary of the access control methods available to Directory administrators, see Clause 15 of Reference [3].

2.8.1 User Authentication

The purpose of user authentication is to verify the identity of a Directory user, so that access to Directory information can be granted or denied with a high level of confidence that the user requesting the access is in fact who she/he claims to be.

Two authentication schemes are described for the Directory, *Simple Authentication* and *Strong Authentication.*

2.8.1.1 Simple Authentication

The simple authentication procedure involves passing a user's name, in the form of the DN of the user's Directory entry, and a password to the Directory. The Directory checks for an entry with the supplied DN and, if such an entry exists, compares the supplied password against the value stored in the **UserPassword** attribute of the entry. If the value matches, the user is authenticated. This method offers a minimum security approach, since neither the user's name nor the password are encrypted or encoded in any way.

Simple authentication can be made somewhat more secure by using a one-way hashing function on the user's name and password supplied to the Directory. This function involves the user's name, password, time stamp, and a collection of random numbers.

2.8.1.2 Strong Authentication

The strong authentication scheme adopted by the Directory standard relies on the use of a public-key cryptosystem, in which each user possesses two keys, one public and one private, the latter of which is known only to the user. Each of these keys may be used to encipher or decipher the user's authentication information, in a complementary fashion (i.e., if the information was enciphered with the private key, it must be deciphered with the public key, and vice versa). If a user's public key is held by the Directory, then it can be used to confirm the user's identity if the user submits his/her authentication information encrypted using his/her private key.

2.8.2 Access Control

In order to control access to Directory information, the DIB is viewed as a collection of *protected items*:

- *Entries;*
- *Attributes;*
- *Attribute Values;* and
- *Names.*

Each protected item has associated with it a set of permissions, representing the access rights of users, groups of users, or the general public. These permissions are further broken down on the basis of the Directory operations.[19]

The permission categories associated with entries (and names) are:

- **Read.** The entry contents may be read only if the entry is explicitly named in the operation;
- **Browse.** The entry contents may be accessed without the entry being explicitly named in the operation;
- **Add.** An entry may be created;
- **Remove.** The entry may be deleted;
- **Modify.** The contents of the entry maybe modified, provided that appropriate permissions exist on the attribute and attribute value level;
- **Rename.** The entry's RDN may be changed;
- **Disclose On Error.** The entry's name may be disclosed if an error occurs;
- **Export.** Permits the entry to be moved to a new location in the DIT, using the ModifyDN operation (see sec. 2.6);
- **Import.** Permits an entry to be removed from another location and placed in the location where the permission applies; and
- **Return Distinguished Name.** Allows the DN of the entry to be disclosed in an operation result.

The permissions associated with attributes and attribute values are slightly different than those for entries:

- **Compare.** Attributes and values may be compared to supplied values;

[19] See Section 2.6 for a list of the Directory operations.

- **Read.** The attribute or value may be returned in response to a read or search operation;
- **FilterMatch.** The attribute or value may be used in the evaluation of a search filter (see sec. 2.6);
- **Add.** Permits the addition of an attribute or value;
- **Remove.** Permits the removal of an attribute with all its values, or the removal of a single attribute value; and
- **Disclose On Error.** Permits the presence of an attribute or attribute value to be disclosed in case of error.

For each permission category, each protected item has an indication of which users or groups of users possess that permission (or whether the permission is publicly available). Thus, when a user requests a particular operation, the Directory locates the protected item(s) in question and ascertains whether the user has permission before carrying out the operation. If not, then the operation is not carried out and a security error may be returned, depending on the disclosure permission for the protected item in question.

2.9 Operational Bindings

An *Operational Binding* is the formalization of an agreement between two Directory Administrations for their respective DSAs to provide services to, or receive services from, one another on an exclusive basis. The operational binding framework is deliberately left very general, so that it can serve as the framework for many different kinds of operational binding agreements which can be envisioned in the future. For the time being; however, the principal use for operational bindings is the setting up, modification and termination of Shadowing Agreements (see sec. 2.10).

An operational binding has the following key components:

- The two DSAs between which the binding will exist. The binding may be symmetric, with each DSA providing the same set of services and having the same role in the management of the operational binding, or asymmetric, in which case each DSA will provide different services (e.g., as in the Shadow Supplier and Shadow Consumer in the Directory Replication model) and may have differing roles in the management of the operational binding;
- An agreement describing the services the DSAs will provide to one another. This agreement is made between the Administrative Authorities responsible for the DSAs involved, perhaps as a legal contract or inter-organizational memo;
- The set of Directory operations to be used by the DSAs involved to carry out the services defined by the operational binding. These operations embody the content of the operational binding agreement into a form which is readily machine interpretable and can thus be used as the basis for protocol exchanges between the two DSAs within the context of the operational binding;
- Operations for the management of the operational binding, providing for its establishment, modification and termination; and

- An identifier for the operational binding which is unique between the two DSAs, such that this identifier, plus the names of the DSAs, constitute a globally unique identifier for the operational binding.

The Directory Standard (Reference [1]) establishes a fourth protocol, the *Directory Operational Binding Management Protocol*, or DOP, to furnish Directory operations for the management of operational bindings. The protocol defines operations for the establishment, modification and termination of an operational binding.

To illustrate how operational bindings are used in the setting up of replication agreements between DSAs (see sec. 2.10) suppose, in Figure 2.11 of Section 2.7, that an agreement was made between the Administrative authorities responsible for DSAs B and C stating that DSA B should keep a shadow copy of naming context D, to be updated at set intervals. Further suppose that the management of the operational binding itself is to be symmetrical, i.e., either DSA may establish, terminate or modify any instance of the binding.

The operational binding agreement is specified using the **OPERATIONAL-BINDING** information object class, as defined in Clause 23.3.1 of Reference [3]. This specification is referenced whenever a DSA performs a management operation on the operational binding. The shadowing agreement is brought into effect when either DSA establishes an instance of the operational binding specified for it. Once in effect, the operations of the Directory Information Shadowing Protocol (DISP) are used to carry out the functions associated with the shadowing agreement, until the operational binding specifying the shadowing agreement is either modified or terminated by either Directory System Agent. Thus the DOP and the DISP are used in concert in order to define a shadowing agreement and carry out the operations necessary to maintain it.

2.10 Replication

Replication is a mechanism used within the Directory whereby information stored in one part of the Directory may also be stored elsewhere as one or more copies. While this mechanism is of minimum visibility to the Directory user, it has the advantages that it increases efficiency of information retrieval and decreases response time to operation requests by placing frequently accessed information "closer" to the user.

For example, in a situation where a particular DSA is either heavily utilized or in a remote location (perhaps Europe or Asia) or both, it may be advantageous to make a copy of the DIT subtree we are interested in on the remote system and import it to our local Directory System Agent. This reduces the load on the remote DSA and improves the Directory's performance from our point of view, because our access to the data is much faster.

Replication also provides an extra degree of reliability and robustness to the Directory in that, if one of the systems holding a copy of an entry becomes inoperable, the information may still be retrieved from another system.

Replication carries with it some potential pitfalls as well as advantages, in that the replicated information may not be up-to-date, and inconsistencies may exist between the replicated data and the master copy. How long these inconsistencies persist will depend on the replication agreement that has been set up between the DSAs concerned – the copies may be updated immediately upon

update to the master, or periodically, perhaps daily, weekly, etc. However, the user always has the option to request information directly from the master copy, making the most current version of the information available if necessary.

2.10.1 Shadowing and Caching

The Directory Standard describes two kinds of replication, namely *Shadowing* and *Caching*. Caching encompasses any non-standardized method for replicating data. For instance, a Directory User Agent (DUA) may simply make a copy of every entry that passes through it, and submit that copy in response to a query about the entry, rather than contact the Directory to request the information. It is easy to see how DSA products might also do this with regard to entry information stored in remote Directory System Agents.

Shadowing, the standard form of replication, sets up replication agreements between DSAs specifying which parts of the DIT are to be replicated, how often copies are to be updated, whether the copies, in part or in whole, may be copied out to other DSAs, the address of the DSA holding the master copy, and so on. These agreements are set up via *Directory Operational Bindings*, which are discussed in Section 2.9.

2.10.1.1 Shadowed Information

The replicated Directory information is always in the form of a contiguous subtree taken from within a single naming context on the master DSA, known as the *Shadow Supplier*. The unit of replication[20] in shadowing is termed *Shadowed Information*, which encompasses the replicated information (referred to as the *replicated area*) as well as information about its origin, extent, etc. The shadowed information has three components:

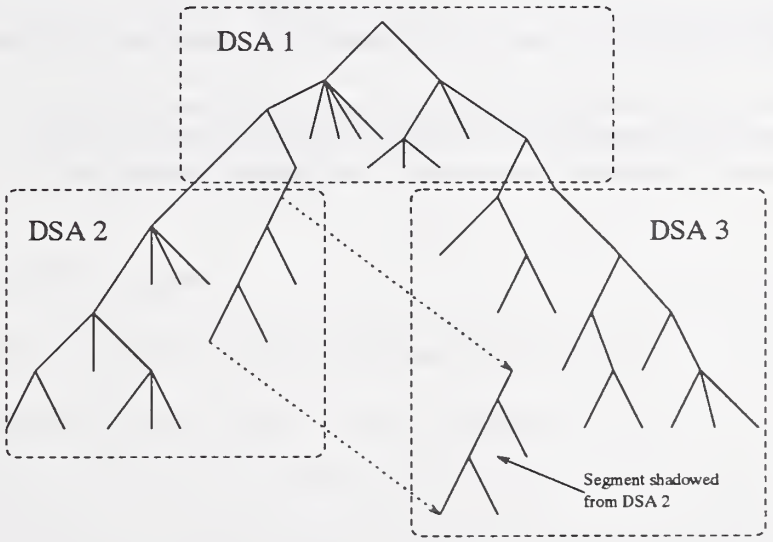


Figure 2.12: *An example showing shadowing of a fragment of the Directory Information Tree.*

[20]For the formal specification of the unit of replication, see Clause 9.2 of Reference [10].

- **Prefix information.** Information regarding the root entry of the replicated subtree, known as the *replication base entry*;
- **Area information.** Information about the Directory entries contained in the replicated area; and
- **Subordinate information.** References to naming contexts subordinate to the replicated area.

In Figure 2.12, DSA 2 acts as a Shadow Supplier to DSA 3, the Shadow Consumer. The DSA 3 now maintains its own local copy of the Shadowed Information.

2.10.2 Shadow Operational Services

The updating of shadowed information is accomplished via three Directory operations which constitute the *Directory Information Shadowing Protocol*, or DISP: *Coordinate Shadow Update, Request Shadow Update* and *Update Shadow.* Through these services, shadowed information can be updated at the instigation of either the shadow supplier or shadow consumer.

- **Coordinate Shadow Update.** This service is used by the shadow supplier to indicate which shadowed information it intends to update, and how. It is issued in response to a *Request Shadow Update* operation or subsequently to an *Update Shadow* operation;
- **Request Shadow Update.** This service is issued by the shadow consumer (i.e., the DSA holding replicated information) in order to request an update of the shadowed information; and
- **Update Shadow.** This service is used by the shadow supplier to send out updated shadow information to a shadow consumer. It is invoked only after one of the previous services.

Each exchange carries information about the shadowing agreement to which it refers, the type of update to be performed (complete, incremental or no change), and so on.

Chapter 3

Functional Evaluation of X.500

3.1 Mandatory Functions

In order to be considered a candidate for procurement, an X.500 product must satisfy the criteria set out in this section. These criteria are derived from the International Standards Organization (ISO) 9594 set of standards (technically aligned with the CCITT X.500 series of recommendations), Reference [1], the Industry/Government Opens Systems Specification, Reference [11], and the NIST Stable Implemention Agreements for Open Systems Interconnection Protocols, Reference [12].

Since the Directory is composed of two distinct components, the Directory System Agents and the Directory User Agents (DSAs and DUAs), requirements for each type of component are set out separately. The DSAs are further subdivided into all DSAs, Shadow Supplier DSAs and Shadow Consumer Directory System Agents.

For each component, the requirements fall into three categories: *Statement Requirements*, which describe the properties and capabilities of a product which must be stated by vendors; *Static Requirements*, which describe invariant, mandatory operational requirements placed on the the product; and *Dynamic Requirements*, which describe operational requirements placed on the product which may change according to circumstances.

3.1.1 Directory User Agents

3.1.1.1 Statement Requirements for Directory User Agents

The manufacturer of a DUA product must state the following:

Which of the Directory Access Protocol operations conformance is claimed for;

The authentication capability for which conformance is claimed (none, simple or strong);

Which of the following extensions the DUA is capable of initiating and for which conformance is claimed:

1. **subentries.** If this service control is set, then **Search** and **List** operations shall access only subentries. If not set, these operations shall access only normal entries;

2. **copyShallDo.** This service control indicates that a query need not be chained if it can be partly satisfied from a local copy of an entry;

3. **attributeSizeLimit.** This service control indicates the maximum permissible size of an attribute for it to be returned in response to a query. If an attribute exceeds this size, it is omitted from the result, and a flag indicating that the entry result is incomplete is set;

4. **extraAttributes.** This component of the **EntryInformationSelection** type (used in **Read** and **Search** requests to specify which information should be returned) is used to specify additional user or operational attributes to be included in the result;

5. **modifyRightsRequest.** This component of the **Read** argument is used to request the return of the requestor's modification rights to the entry and its attributes;

6. **pagedResultsRequest.** This component of the **List** argument specifies whether results of the operation are to be returned page by page;

7. **matchedValuesOnly.** This component of the **Search** argument specifies that attribute values which did not not match specifically with values in the search filter shall not be returned;

8. **extendedFilter.** This component of the **Search** argument specifies an alternative search filter for use by 1994-conformant systems;

9. **targetSystem.** This component of the **Add Entry** argument specifies the DSA which shall hold the new entry;

10. **useAliasOnUpdate.** This describes a bit in the **criticalExtensions** element of **CommonArguments**, the set of arguments common to all operations.[21] If this bit it set, and the **dontDereferenceAlias** element of **CommonArguments** is not set, alias entries will be dereferenced during the commission of an **Add Entry** operation; and

11. **newSuperior.** In a **ModifyDN** operation argument, this component specifies the name of an object entry in the DIT which is to become the new superior entry of the entry subject to the **ModifyDN** operation.

3.1.1.2 Static Requirements for Directory User Agents

The Static Requirements for DUAs are as follows:

Support for the **Bind** and **Unbind** operations is required;

The DUA product must be able to the support the **directoryAccessAC** application-context, as defined in Clause 7 of Reference [6]. This sets out the protocols and abstract syntaxes to be used during a Directory Access session;

The DUA product must be able to handle the following errors: **Abandoned, AbandonFailed, AttributeError** and **NameError**;

The product must be capable of handling the extensions to which conformance was claimed in Section 3.1.1.1, and in any case where the product functions as an IGOSS Administrative DUA

[21] See Section 2.6.9.

(i.e., a DUA capable of supporting all DAP operations), the following extensions must be supported: **extraAttributes, subentries, newSuperior** and **useAliasOnUpdate**;

The product must support the character set requirements set out in Part 11, Clause 7.1.1 of Reference [12], i.e., **T.61 String, PrintableString, NumericString** and **UNIVERSAL STRING**;

The product must support None and ID-only/Simple Uncorroborated modes of authentication. In the former method, no verification of the user's ID is carried out at all, while ID-only is the simple passing of a user ID, which is checked via a compare operation by the DSA against a list of legitimate users; and

The product shall have the capability to perform the normalization of protocol elements containing the **UTC Time** element according to the rule specified in Part 11, Clause A.4.1 of Reference [12]. This process consists of filling in the seconds field with "00" whenever this field has been omitted, and converting the string to the "Z" form.

3.1.1.3 Dynamic Requirements for Directory User Agents

The DUA product must satisfy the following Dynamic Requirements as set out in Reference [6]:

The product must conform to the mapping onto used services defined in Clause 8 of Reference [6]. This mandates the mappings between the X.500 services and those of the supporting Open Systems Interconnection (OSI) services (Association Control Service Element (ACSE), Remote Operation Service Element (ROSE), Presentation, Session, and so on); and

The product must conform to the rules of extensibility procedures defined in Clause 7.5.1 of Reference [6]. These rules deal with the interoperation of products adhering to different versions of the Directory standard.

3.1.2 Directory System Agents

3.1.2.1 Statement Requirements for Directory System Agents

The manufacturer must state the following with regard to a DSA product:

The Application Contexts for which conformance is claimed: **directoryAccessAC, directorySystemAC** and **directoryOperationalBindingManagementAC**. If knowledge of a DSA has been disseminated to other DSAs, then it shall claim conformance to the **directorySystemAC**. Conformance must be claimed at the Application Context level and shall not be claimed to individual operations;

The operational binding types for which conformance is claimed: **shadowOperationalBindingID, specificHierarchicalBindingID**, and **non-specificHierarchicalBindingID**;

Whether the DSA is capable of acting as a first level Directory System Agent;

If conformance is claimed to the **directorySystemAC**, whether or not the chained mode of operation is supported;

The security level(s) for which conformance is claimed (none, simple, strong);

The selected attribute types, as defined in Reference [7], and any other attribute types, for which conformance is claimed and whether, for attributes based on the syntax **DirectoryString**, conformance is claimed for the **UNIVERSAL STRING** choice;[22]

The operational attribute types defined in Reference [3] and any other operational attribute types for which conformance is claimed;

The selected object classes, as defined in Reference [8], and any other object classes, for which conformance is claimed;

The extensions listed in Table 1 of Clause 7.3.1 of Reference [4] that the DSA is capable of responding to for which conformance is claimed;

Whether conformance is claimed for the return of alias names;

Whether conformance is claimed for indicating that returned information is complete;

Whether conformance is claimed for modifying the object class attribute to add and/or remove values identifying auxiliary object classes;

Whether conformance is claimed to Basic Access Control;

Whether the DSA is capable of administering the subschema for its portion of the Directory Information Tree;

The selected name bindings, and any other name bindings, for which conformance is claimed; and

In accordance with Reference [12], the DSA manufacturer shall state to which of the following conformance classes the product belongs:

- **0:** Centralized Directory System Agent. Supports only the **directoryAccessAC**; or
- **1:** Distributed Directory System Agent. The DSA shall implement all operations in the Application Service Elements forming part of the Application Contexts for which it claims conformance. The DSA shall support the **directoryAccessAC** and may optionally support the **directorySystemAC**.

3.1.2.2 Statement Requirements for Shadow Supplier Directory System Agents

In addition to the general statements described above, the manufacturer of a DSA product claiming conformance as a shadow supplier must state the following:

The application contexts for which conformance is claimed as a shadow supplier: **shadowSupplierInitiatedAC, shadowConsumerInitiatedAC, reliableShadowSupplierInitiatedAC,** and **reliableShadowConsumerInitiatedAC**; and

To which degree the **UnitOfReplication** is supported. Specifically, which (if any) of the following optional features are supported:

- Entry filtering on **ObjectClass**;

[22] It is a static requirement that all selected attribute types as defined in Clause 5 of Reference [7] should be supported.

- Selection/Exclusion of attributes via **AttributeSelection**;
- The inclusion of subordinate knowledge in the replicated area; or
- The inclusion of extended knowledge in addition to subordinate knowledge.

3.1.2.3 Statement Requirements for Shadow Consumer Directory System Agents

In addition to the general statements described above, the manufacturer of a DSA product claiming conformance as a shadow consumer must state the following:

The application contexts for which conformance is claimed as a shadow consumer: **shadowSupplierInitiatedAC**, **shadowConsumerInitiatedAC**, **reliableShadowSupplierInitiatedAC**, and **reliableShadowConsumerInitiatedAC**;

Whether the DSA can act as a secondary shadow supplier; and

Whether the DSA supports shadowing of overlapping units of replication.

3.1.2.4 Static Requirements for Directory System Agents

In addition to the applicable Statement Requirements, every DSA package must satisfy the following Static Requirements:

It must have the capability to support the application contexts for which conformance is claimed: all DSAs shall support **directoryAccessAC** or **directorySystemAC** or both;

It must have the capability to support the information framework defined by its abstract syntax in Reference [3];

It must conform to minimal knowledge requirements defined in Reference [3], i.e., each non-first level DSA shall maintain a single superior reference; each DSA that is the master DSA for a naming context shall maintain subordinate or non-specific subordinate references to DSAs holding naming contexts immediately subordinate to that naming context;

If conformance is claimed as a first-level DSA, the product must conform to the requirements for support of the root context;

The DSA must have the capability to support the attribute types for which conformance is claimed, as defined by their abstract syntaxes, and in any case shall support the selected attribute types defined in Reference [7] and their associated attribute syntaxes;

It must have the capability to support the object classes for which conformance is claimed, and in any case the DSA shall support all selected object classes defined in Reference [8];

The DSA shall support all matching rules defined in Clause 7 of Reference [7];

The product must conform to the extensions for which conformance was claimed;

If the capability to administer subschema, as defined in Reference [3], is claimed, the DSA shall be able to do this administration;

The DSA shall support collective attributes, and must have the capability to perform the related procedures, as outlined in Reference [4];

The DSA shall support hierarchical attributes, and must have the capability to perform related procedures, as outlined in Reference [4];

The DSA must have the capability to support the operational attribute types for which conformance is claimed;

If conformance is claimed to Basic Access Control, the DSA must have the capability to hold Access Control Information (ACI) items conforming to the definitions of Basic Access Control;

The DSA shall support Simplified Access Control (SAC). A class 2 subtree specification, as described in Part 11, Clause 8.10 of Reference [12] shall be supported;

The DSA shall support operational attributes associated with the supported access control scheme(s), and the **createTimestamp, modifyTimestamp, creatorsName** and **modifiersName** attributes;

The DSA shall support the **UserPassword** attribute, described in Reference [9];

A DSA which supports any kind of strong authentication shall support the **strongAuthenticationUser** and **certificationAuthority** attributes, as defined in Reference [8];

A DSA supporting any form of strong authentication shall support the following attribute types defined in Reference [9]: **UserCertificate, CACertificate, CrossCertificatePair,CertificateRevocationList** and **AuthorityRevocationList**;

The DSA shall be configurable to allow new attribute types to be defined by the DSA administrator. Extensibility of supported attribute types shall include the following features:

- New types may be defined in terms of syntaxes defined in Clause 6 of Reference [7]; and
- New types may be defined in terms of a new syntax where:

    1. The syntax is one of: **Integer, Null, Boolean, Enumerated, Bit String, Octet String, Object Identifier, Distinguished Name, Case Exact String, Case Ignore String, Numeric String, Printable String, UTC Time** or **Telephone Number**;
    2. The new syntax is an ASN.1 structured type (i.e., **SET, SEQUENCE, SET OF, SEQUENCE OF** and **CHOICE**), possibly including tags, where each component uses one of the syntax forms listed in the previous item; and
    3. The matching rule associated with a locally defined type may be defined using any of the rules described in Clause 7 of Reference [7].

The DSA shall be configurable to allow new object classes to be defined by the DSA administrator. Extensibility of supported object classes shall include the following features:

- New object classes may be defined to be either abstract, structural or auxiliary;
- A new object class may be a subclass of any class described in Reference [8] or it may be a subclass of a locally defined class; and

- A new object class may be defined in terms of any combination of attribute types described in Reference [8] and locally defined attribute types.

The DSA shall be configurable to allow the DSA administrator to define the complete set of allowed name bindings. Each of the name bindings described in Clause 7 of Reference [7] shall be implemented;

The DSA shall adhere to the pragmatic constraints specified in Part 11, Clause 7 of Reference [12]; and

The DSA shall adhere to requirements in Part 11, Annex A of Reference [12] regarding the Maintenance of Attribute Syntaxes.

Distributed DSAs only. The following requirements apply only to Distributed DSAs, as defined in Section 3.1.2.1.

The DSA must be able to carry out name resolution and search continuation for an alias whose dereference points to an entry held outside the DSA, as described in Part 11, Clause 9.1.5 of Reference [12];

The DSA must be able to carry out simple authentication of a user whose entry is outside the authenticating DSA as described in Part 11 of Reference [12], Clause 9.1.7;

The DSA must adhere to requirements regarding the handling of the **TraceInformation** attribute, as specified in Part 11, Clause 9.2.2 of Reference [12];

The DSA must adhere to the requirement regarding propagation of signed arguments specified in Part 11 Clause 9.2.3 of Reference [12];

The DSA must adhere to the requirement regarding referrals and chaining specified in Part 11 Clause 9.2.4 of Reference [12], with the following proviso:

- the conditions under which a distributed DSA does not act on a referral include the following:
  - The **returnToDUA** element of **DSAReferral** indicates the referral is not to be acted on; or
  - Administrative limitations or service policies prevent the DSA from acting on the referral.

The DSA must adhere to underlying services requirements specified in Part 11, Clause 10 of Reference [12];

The DSA shall be capable of supporting the structure and naming rules defined in Reference [8], Annex B;

The DSA shall be able to support all superclasses of supported object classes;

The DSA shall be able to support the storage and use of attribute type information, as defined in the Reference [7], including their use in naming and access to entries;

The DSA shall support the encoding, decoding and matching of all the attributes in the Naming Prefixes of every naming context they hold (see Reference [5], Clause 9);

The attributes and attribute sets in Reference [7], associated with the object classes listed below are required. The storage and use of the object classes below is also required:

Top, DSA, Alias, Country, Locality, Application Process, Organization, Organizational Unit, ApplicationEntity,Device, Group of Names, OrganizationalPerson, OrganizationalRole, ResidentialPerson

An object class may not be defined as a subclass of itself;

The DSA must support all character sets and/or other nameforms defined in Reference [7], including **T.61, PrintableString** and **NumericString**;

It is a minimum requirement that invoke Application Protocol Data Units (APDUs) and return result APDUs shall be accepted unless their size exceeds 2**18 - 1 (262,143) octets;

At minimum, 8 nested **Filter** parameters will be supported, with a total limit of 32 **FilterItems**;

All distributed DSAs shall be capable of acting as a holder and a propagator of Directory information;

The DSA shall support the **Versions** component of the Bind argument;

The DSA shall support the Reference [12] Directory Common Application Directory Profile;

The DSA shall be able to hold and use the following reference types:

- **superior:** Non-first level DSAs shall have precisely one. First level DSAs have none;
- **subordinate:** Mandatory;
- **cross:** Mandatory; and
- **non-specific subordinate:** Optional.

A first level DSA shall be able to hold and use the root context[23] and shall hold as master at least one naming context immediately subordinate to the root of the Directory Information Tree;

If the DSA supports the **directorySystemAC** it must be able to accept a chained request and generate a referral;

In order to perform simple authentication of a user whose entry potentially resides on another DSA, a DSA must be able to invoke DSA **Compare** and **Read** operations, and must thus support the **directorySystemAC**;[24] and

If the propagation of a **Search** operation involves request decomposition, the **traceInformation** argument of the original Search request shall not be reset, rather the full **traceInformation** for the overall Search to the point where one or more new Search requests are generated shall be included in the new Search requests;

[23] I.e. a first-level DSA shall act as if it holds the (hypothetical) root of the Directory Information Tree.

[24] In order to perform simple authentication of a user whose entry potentially resides on another Directory System Agent.

Centralized DSAs only. The following requirements apply only to Centralized DSAs, as defined in Section 3.1.2.1.

The DSA must not implement shadowing.

3.1.2.5 Static Requirements for Shadow Supplier and Consumer Directory System Agents

In addition to the general statement requirements, the statement requirements for Shadow Supplier DSAs, the statement requirements for Shadow Consumer DSAs and the general static requirements, Shadow Supplier and Consumer DSA packages must satisfy the following static requirements:

A DSA shall, at a minimum, support either the **shadowSupplierInitiatedAC** or the **shadowConsumerInitiatedAC**. If the DSA supports the **shadowSupplierInitiatedAC**, it may optionally support the **reliableShadowSupplierInitiatedAC**. If the DSA supports the **shadowConsumerInitiatedAC**, it may optionally support the **reliableShadowConsumerInitiatedAC**;

Each shadow supplier DSA shall maintain a consumer reference for each shadow consumer DSA that it supplies with a replicated area;

Each shadow consumer DSA shall maintain a supplier reference for each shadow supplier DSA that supplies it with a replicated area;

The DSA must support the **modifyTimeStamp** and **createTimeStamp** operational attributes;

The DSA must provide support for the **copyShallDo** service control;

The DSA must implement the **Directory Information Shadowing Protocol (DISP)** as described in Reference [12] Part 11, Clause 8.11.[25] A class 2 unit of replication, as described in Reference [12] Part 11, Clause 8.11.3, shall be supported;

All DSAs implementing the **DISP** shall be capable of acting as both shadow supplier and consumer as defined in Reference [10], Clause 3 and shall meet conformance requirements stated in Reference [6], Clauses 9.3 and 9.4;

The DSA shall also support minimum shadowing requirements:

- Support for both **directoryShadowConsumerAC** and **directoryShadowSupplierAC**;
- Support for an **updateMode** whose mode choice includes a specification of **schedulingParameters**; and
- Support for **schedulingParameters** specifications which specify a periodic strategy.

The product supplier shall state which Unit of Replication conformance class is supported:

- **0:** Basic **UnitOfReplication.** The DSA shall be capable of shadowing a unit of replication with the following characteristics:

    1. The area includes a class 0 subtree as defined in Reference [12]; and

[25] See also Section 2.10.

2. The area includes a specified **knowledgeType** (e.g., master, copy or both).

- **1:** Intermediate **UnitOfReplication.** The Directory System Agent shall fully support Basic **UnitOfReplication** and shall also be capable of shadowing a unit of replication with the following characteristics:

  1. The area includes a class 1 subtree as defined in Part 11, Clause 8.10 of Reference [12]; and
  2. The knowledge includes an **extendedKnowledge** element with value TRUE.

- **2:** Maximal **UnitOfReplication.** The DSA shall fully support class 1 and shall be capable of shadowing a unit of replication whose specification uses **AttributeSelection** (including selection on class). The DSA shall be capable of supporting overlapping replicated areas as described in Reference [10], Clause 9.2.5.

3.1.2.6 Dynamic Requirements for all Directory System Agents

All DSAs must satisfy the following dynamic requirements:

Conform to mapping onto used services defined in Clause 8 of Reference [6];

Conform to procedures for distributed operation of the Directory related to referrals, as defined in Reference [5];

If conformance is claimed to the **directoryAccessAC** application context, conform to the procedures of Reference [5] as they relate to the referral mode of the Directory Access Protocol;

If conformance is claimed to the **directorySystemAC** application context, conform to the referral mode of interaction, as defined in Reference [5];

If conformance is claimed to the chained mode of interaction, conform to the chained mode of interaction, as defined in Reference [5];

Conform to the rules of extensibility defined in Clause 7.5.2 of Reference [6];

If conformance is claimed to Basic Access Control, have the capability of protecting information within the DSA in accordance with the procedures for Basic Access Control;

If conformance is claimed to the **shadowOperationalBindingID**, conform to the procedures of Reference [10] and Reference [3] as they relate to the Directory Access Protocol;

If conformance is claimed to the **specificHierarchicalBindingID**, conform to the procedures of Reference [5] and Reference [3] as they relate to specific hierarchical operational bindings;

If conformance is claimed to the **non-specificHierarchicalBindingID**, conform to the procedures of Reference [3] and Reference [5] as they relate to non-specific hierarchical operational bindings; and

The DSA shall adhere to requirements to support Session Version 2 as described in Part 11, Clause 10.2 of Reference [12].

3.1.2.7 Dynamic Requirements for Shadow Supplier and Shadow Consumer Directory System Agents

All Shadow Supplier and Shadow Consumer DSAs must satisfy the following dynamic requirement:

Conform to the procedures of Reference [10] as they relate to the **Directory Information Shadowing Protocol**.

3.2 Non-Standard Functions

Section 3.1 describes the features and capabilities which an X.500 Directory product *must* have in order to satisfy current U.S. federal government procurement guidelines. However, these features and capabilities, though essential, provide only the bare skeleton of functionality required for an efficient and usable Directory system.

This section sets out to explore some non-mandatory, sometimes even non-standard features [26] which can enhance the performance and usability of the Directory as an information storage and retrieval tool. As in Section 3.1, Directory User Agents and Directory System Agents will be treated separately, except in the critical area of interoperability, to which a specific section is dedicated.

3.2.1 Interoperability

Interoperability between two Directory components means that they should be able to exchange Directory operation requests, results and errors without error and with a mutual interpretation of the various parameters and their values which appear in the protocol exchanges. Interoperability between two instances of the same product is usually taken for granted, so the term usually applies to the inter-working of two or more implementations from different vendors, or two or more different implementations from the same vendor. While it may seem self-evident at first that two implementations of the same standard should inevitably work together, experience has shown that this is rarely the case, at least not initially. Despite the careful language of the standards themselves and the work put into the production of various sets of Implementors' Agreements[27] and standards profiles, ambiguities often exist, and different interpretations on the part of software production staff of different vendors invariably occur.

Various guides to X.500 product interoperability exist: the *OSINET Stable Interoperability Tests* document, Reference [14], gives the results of interoperability testing performed between different X.500 products in a controlled environment. At the time of writing, the Joint Interoperability Test Command (JITC) of the Defense Information Systems Agency (DISA) of the Department of Defense (DoD) is also performing X.500 conformance and interoperability tests and recording the results in a publicly accessible register. If the product(s) under consideration is (are) not included in these sources, then the manufacturer should be consulted as to whether the product will interoperate with all the other systems under consideration, and should be held accountable to any claims made.

Interoperability between different components of an organization's Directory system is always highly

[26] Non-standard in the sense that the features lie outside the scope of the Standard.

[27] See, for example, Reference [12].

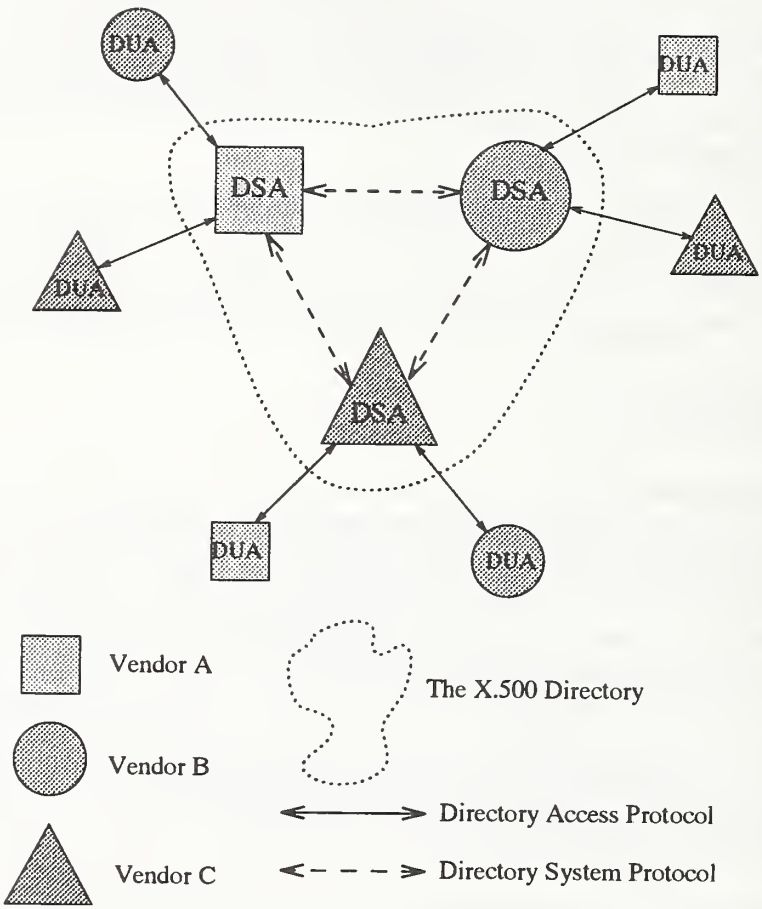


Figure 3.1: *Interoperability in the X.500 Directory.*

desirable, but there are two situations in which it becomes critical: when policy or circumstances dictate that the Directory will be made up of heterogeneous components (i.e., components from different manufacturers; different products from the same manufacturer; different releases of the same product line from the same manufacturer; and so on); and when policy or circumstances dictate that the organizational Directory must communicate with other X.500 installations outside the organization, which in all probability will use software and/or hardware which differs in some way from that in use locally.[28]

All DUAs should be able to interoperate with any DSA component in the organization's local Directory. All DSAs should be capable of interoperating with all DUAs and all DSAs in the organization. Figure 3.1 gives a schematic rendition of this idea.

3.2.2 Directory User Agents

Many of the features considered in this section apply to the human user interface to the DUA, as opposed to the protocol engine component, but the two components are treated together here since the Directory Standard regards the user interface as an integral part of the DUA (see Reference [3], Clause 6.2, Note 3). Bearing in mind the potentially huge range of possibilities for the presentation and manipulation of Directory information, it is only possible to discuss the most general principles here, and, even so, possibilities may be missed.

Six general principles, which can be roughly grouped into three pairs, will be used as the framework for exploring the realization of the DUA's potential. *Configurability* and *Ease of Integration* enable the product to be tailored to best suit the user's requirements while creating minimal disturbance to the established user environment. *Accessibility* of the information held in the Directory and *Clarity* in the presentation of this information are vital if the service is to be of genuine use, providing ease of access to needed information. Finally, *Ease of Operation* and *Informativeness* deal with the need for the product to enhance rather than impede the user's access to information, providing an easily understood interface with detailed and informative help pages at each step.

3.2.2.1 Ease of Configuration

One quality which is highly desirable for all features of the DUA is that they be easy to configure. In addition, reconfiguration of any feature should cause a minimum of disturbance to the system. For example, no rebooting or restarting of the software should be necessary in order to reconfigure a feature: the operation of the DUA should be essentially undisturbed. Also, the reconfiguration should go into effect immediately after it is complete, and be applied to subsequent operations. However, results of any Directory operations initiated prior to the reconfiguration may need to be treated in accordance with the configuration in place when the corresponding operation request was issued.

Additionally, it should be possible to configure each feature from outside the DUA itself (i.e., when the DUA is not running, or as the enactment of a system-wide configuration policy) by editing some configuration file, using either a regular text editor or a tool specialized for the task (preferably both), which is judged to be the best suited to the local user community.

[28] As well as differences between the X.500 products themselves, there may be differences in the underlying communications software. Such differences are not addressed here.

Examples of features that it should be possible to configure in accordance with the principles laid out above include:

- **Common Arguments.** The Common Arguments[29] associated with each Directory Operation are often relatively constant for the duration of a session, and as such may be inserted into each operation automatically by the DUA, without the need for the user to enter them individually. They should be easy to configure and easily modified. For example, the system

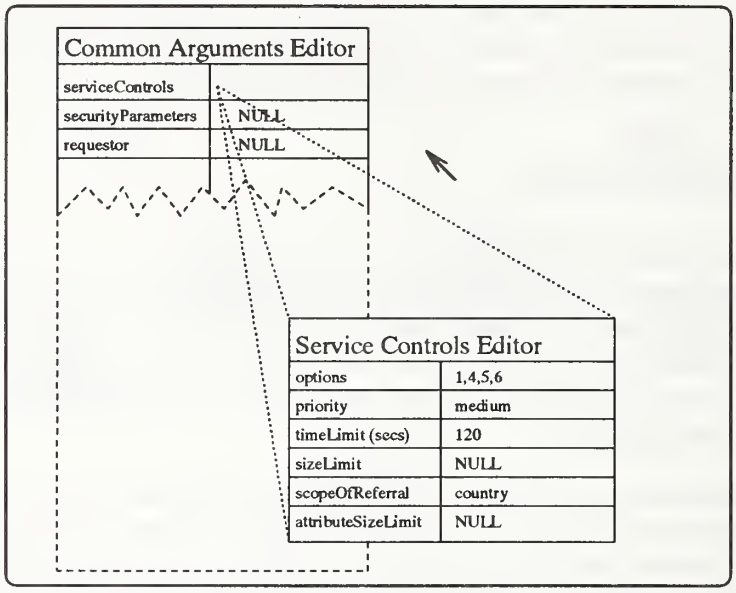


Figure 3.2: *An example of a menu-based editor for* Common Arguments.

might provide a default Common Arguments configuration file for each user, which is easily modified at any time, either during a session by using a specialized tool such as an editable pop-up menu, or outside sessions by simply editing the file with a normal text editor or special tool provided for the purpose. Figure 3.2 shows a simple example of a tool for editing the Common Arguments parameter.

- **Local Caching Capability.** It is possible for a DUA to cache Directory information by simply storing any information it receives from a Directory System Agent. Such caching is outside the scope of the Standard, and it is not possible to set up an agreement between DUA and DSA to ensure updates to the cached information, etc., but local caching can nevertheless prove to be a useful tool and can help increase retrieval speed for the user while decreasing network traffic and DSA workload. Local caching algorithms can vary from the most basic (e.g., store all information received for a designated time period, indexing by distinguished name) to the more elaborate. For example, if a certain entry is accessed more than a threshold number of times during a session, the DUA may be configured to send out an *independent* **Read** operation to obtain all possible information from the entry in question, and use the cached information to satisfy subsequent queries about the entry. In this latter case, the DUA is acting "in the background," independently of the user, to obtain information that it is assumed the user is likely to need.

[29] See Section 2.6 for more on Common Arguments.

- **Specification of Nonstandard Schema Elements.** It is regarded as essential that the user or administrator of a DUA be able to define nonstandard schema elements designed to accommodate local requirements. For the DUA, these elements are attribute types, object classes and DIT content rules. Ideally, these can be specified either with a regular text editor, using specified formats, or by using specialized tools provided with the DUA software. The definition mechanisms should incorporate the notions of attribute hierarchies, object class (multiple) inheritance, and so on. Any tools provided will likely be windows-based for clarity of information presentation.

- **Default DSA Specification.** It should be possible for a user to specify a "default" DSA with which to bind when initiating a Directory session, possibly along with a list of backup DSAs to contact, in order of preference, if the preferred DSA is unavailable for some reason. This avoids the need for the user to specify a particular DSA on each bind attempt, and thus saves time and effort, while furthering the notion that the user is connecting to the Directory as a whole, as opposed to a single server.

  Alternatively, a list of possible DSAs to contact, labelled with user-friendly names, could be displayed, enabling the user to simply click on the desired name in order to bind to the Directory.

- **Default "Home Position".** Such a feature would enable the user to configure an entry in the Directory which would be the default position at which all operations would be targeted. For instance, this may be the user's own entry, or the naming context prefix for a naming context in which a user or application carries out most processing. Optionally, this entry could be read and displayed when the user binds to the Directory.

- **"Hot list" of Commonly Accessed Entries.** Similar to the default home position, this feature would enable the user to configure a list of commonly accessed entries or entry distinguished name prefixes, which could be brought up on a pop-up menu and easily selected with a mouse, saving the typing of often long Distinguished Names.

- **Full or Abbreviated Attribute Names.** The DUA should be easily configurable for the input and display of full length (e.g., "Organizational Unit") or abbreviated attribute names (e.g., "OU").

- **Confirmation Request.** Before the submission of each operation request to the Directory, the user may be prompted by the DUA interface as to whether the request is ready for submission. The default condition could be set as a toggle.

This is only a selection of features which can and should be easily configurable by the user. In the sections that follow, other features examined in different contexts will also be seen to be clear candidates for user configuration.

3.2.2.2 Ease of Integration

An X.500 DUA product should be easily integrable into the user's existing computing environment, introducing the minimum of additional complexity and presenting an interface which is easily understood and based on clear, open principles. Above all, any DUA package should be known in advance to be capable of interoperating with any DSA package in the organization. Some ways in which a product may achieve these goals include:

- **User Interface.** The user interface of the DUA should be able to run under a variety of widely used, *de facto* or *de jure* standard operating system packages, such as the various windowing environments that are currently available. For clarity of presentation, it is desirable that the DUA use some kind of windowing system, the point here being that it should have the capability to fit easily into whatever system is currently in use at the user site.

- **X.400 Integration.** The principal use of the Directory, at least initially, is envisioned to be as a name to address lookup facility for X.400-based electronic mail systems. With this in mind, it is seen as highly desirable that the DUA provide a programmatic interface or API to such systems. The Directory standard contains various definitions which allow for the storage of X.400 information, and X.400-based systems should be equipped to query the Directory for this information. See also the next item.

- **Standard Application Programming Interface.** The programmatic interface to the DUA should be based on the Portable Operating System Interface (POSIX)/Institute of Electrical and Electronic Engineers (IEEE) API for Directory operations, Reference [13], or some other widely accepted standard. This enables applications developers at the user site to develop applications around a standardized set of function calls in order to Directory-enable their software. It also should minimize the amount of developer effort expended if Directory products are switched, thereby empowering the user community with the ability to select X.500 products on the basis of other, more significant features.

- **Variety of Platforms.** The product should be available to run on a variety of hardware and software platforms. In particular, from the user's point of view, it should be available to run on all platforms in use at the user's site at the time of purchase.

- **Communications Architecture.** The product should be able to accommodate the various types of communications architecture in place at the user site, including pure Open Systems Interconnection (OSI), mixed OSI-Transmission Control Protocol/Internet Protocol (TCP/IP) (i.e., Request For Comments (RFC) 1006, Reference [15]), or whatever other communications architecture may be present.

3.2.2.3 Accessibility of Data

The purpose of the DUA is to obtain information from the Directory and present it to the user. The DUA product should do as much as possible to enhance the accessibility of Directory information; for example, by making it easier for the user to compose the Directory operations necessary for the effective retrieval of the desired information. Some examples of how this might be done are given below:

- **Handling of Referrals.**[30] When a referral is received by the DUA, several courses of action are open to it. The referral information, or just the fact that a referral has been received in connection with a particular operation, might simply be passed on to the user. This information could be accompanied by an option for the user to select whether to proceed with or terminate the operation. Alternatively, the DUA could pursue the referral automatically, possibly flagging the user that it has done so – both these options should be configurable by

[30] See Section 2.6 for an explanation of Referrals.

the user. In the simplest case, the referral may just be discarded and the operation logged as unsuccessful.

- **Automatic Bind.** The DUA might automatically bind to a default DSA on start-up, enabling the user to begin work immediately.[31] This feature should be configurable.

- **Followup List Operation.** The DUA might issue a *List* operation automatically after every **Read** operation to show the subordinates of the entry on which the operation was performed. This information in turn might be used to create a dynamic graphic representation of the DIT in the area in which work is being performed. This feature should be configurable.

- **Tree Build.** The user may wish to see a graphical representation of the DIT at the point at which he/she is working. This might be achieved by the provision of a utility in the DUA which, through the automatic issuance of a succession of **List** requests, can obtain the information necessary to build a graphical representation of the requested DIT fragment. The user might specify the number of levels of the DIT to be diagrammed, relative to a given target entry, both above and below that entry. To take this level of accessibility to the next logical step, Directory operations should be targetable at entries in the displayed DIT fragment by clicking on the displayed entry using the mouse or other pointing device. A default value should be configurable for the number of levels to be displayed, and a system-wide default may obtain. Access controls may prevent a full picture from being displayed.

3.2.2.4 Clarity of Presentation

In order to optimize the usefulness of Directory information to the Directory user, the X.500 product user interface should display all information with the greatest possible clarity. For all practical purposes at the present time, a windows based approach is the most desirable means for the display of information. This enables different information objects or groupings of objects to be displayed as separate entities, drag and drop/cut and paste features, point and click functionality, and so on, minimizing the amount of time spent by the user on such tedious tasks as typing Distinguished Names and repeating operations with slightly different parameters. Some features which might aid in the clarity of presentation of information follow:

- **Pop-up Displays.** Each information item (for example, the contents of a Directory entry, the result of a *Read* operation) is displayed as a list of items in a pop-up window. Each item consists of the attribute type and value(s). When there is more than one value associated with an attribute, or the attribute has a complex syntax, then the entire value or value set is displayed in a new window by clicking on the initial item. This process is recursive for complex elements.

- **Highlighting of Distinguished Values.** Distinguished attribute values might be highlighted or otherwise indicated so that the entry's Relative Distinguished Name can easily be read.

[31] Usually such an automatic bind will grant the user only public access rights in the Directory. The user will need to be re-authenticated if greater access rights are required. The point here is that the DUA is "ready to go" immediately on start-up.

3.2.2.5 Ease of Operation

Ideally, when using the Directory, information recovery is optimized against the effort expended by the user. In other words, the user should be able to recover the maximum information for the minimum time and effort. To this end, the User Interface should include tools to assist in the composition of Directory operation arguments, so that these arguments can be assembled as quickly and easily, and with as few errors as possible. One example of such a tool is the Common Arguments editor mentioned in Section 3.2.2.1. Additional examples of important areas to be addressed by such tools are:

- **Search Filter Editor.** The availability of a tool to enable the user to quickly and accurately build a search filter is highly desirable, since it is through the **Search** operation that most Directory interrogation is likely to take place. Such a tool might permit the user to specify the filter in a way similar to a *Structured Query Language (SQL)* **SELECT** command, or it might allow the user to build the filter in a display window, using cut and pastes from

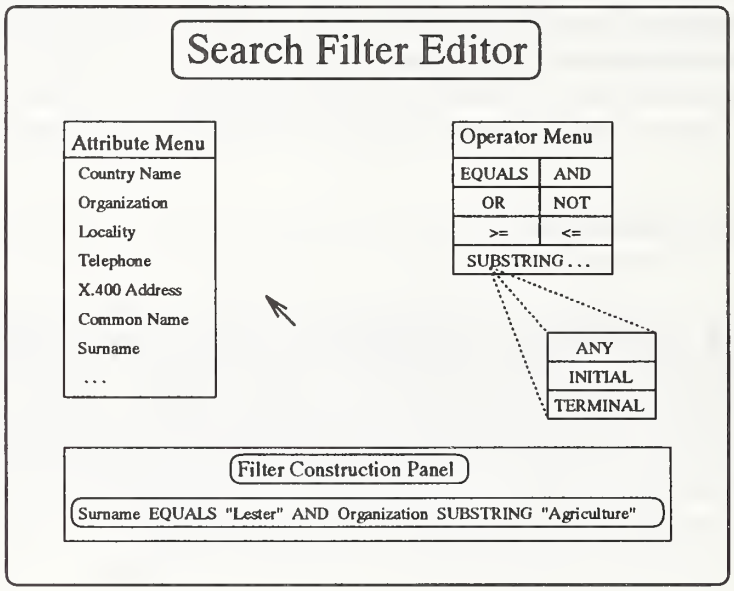


Figure 3.3: *An example of a windows-based Search Filter Editor.*

other windows. These windows would include a pop-up menu of available attribute types, where selecting a type inserts the type into the filter under construction; a similar, smaller, pop-up menu containing the logical connectors for the search filter; and a display window which shows the filter at its present stage of construction (see fig. 3.3). In this way, the user needs only to type the attribute values s/he wishes to use as limiting criteria in the **Search** argument. The tool as described is very general purpose, but this may not always be the optimal approach – see "Operation Template Editor" below.

- **Search Scope.** The **Search** operation is the most powerful interrogation operation available to the user, but it is also the most resource intensive and often the most time-consuming. The provision of a tool to enable the user easily to limit the scope of the search can help to optimize resource usage as well as to get results back to the user more quickly. One method of achieving this might be to provide a mechanism enabling the user to compose a single **Search** request

and have it directed at multiple, small, naming contexts where the user feels the likelihood of a hit will be greatest. Thus, for example, if a user wishes to locate an individual who works in the federal government, and is fairly certain that the person in question works for the United States Department of Agriculture (USDA) or the Environmental Protection Agency (EPA), the tool would send off two essentially identical requests to search the USDA and EPA domains, instead of searching the whole of the U.S. Government domain. This produces a much quicker response time at little extra effort expenditure on the part of the user. While this example might seem trite, the power of such a tool would be much more evident if the number of agencies concerned were larger, or government contractor organizations were involved, or the search scope in each instance were more restricted (for example "personnel office").

- **Entry Modification Editor.** Whenever the user indicates an intention to modify an entry, the interface might automatically retrieve the contents of the entry using a **Read** operation, and display them in an editable window. The user can then make any desired modifications to the contents of the entry with the editor, the modifications are automatically converted into a *Modify Entry* request and shipped off to the Directory.

- **Operation Template Editor.** In many situations, exposure of the DUA user to the full potential of a DUA is neither necessary nor desirable. In such cases, it may be desirable to provide the user with a custom DUA which provides operation templates; for example, a template which offers a name search facility, such as that shown in Figure 3.4, in which the

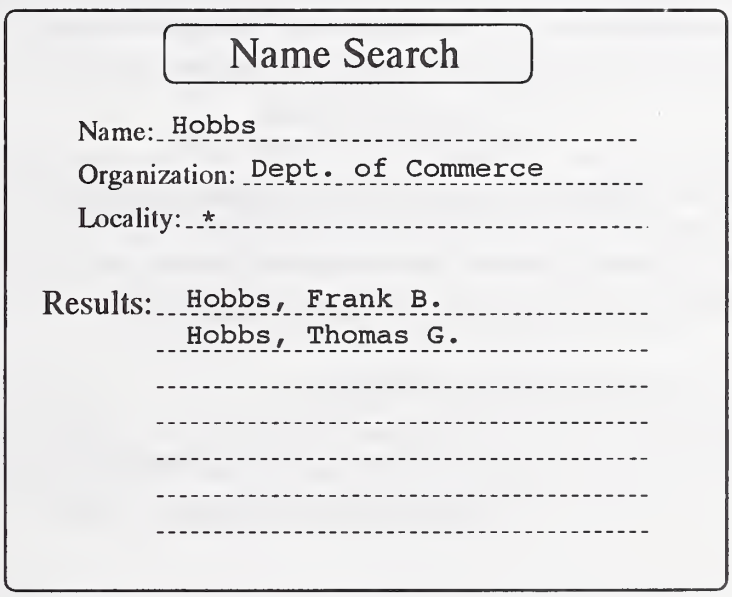


Figure 3.4: *An example of an operation template for a Name-based Search operation.*

user needs to supply only a common name and possibly one or more of a restricted number of other parameters, such as organization name, locality, and so on. The provision of such simplified operation templates will be tailored to the needs of the particular organization, and templates will be designed to accommodate those operations most commonly required by users. In particular, such templates can isolate the user completely from exposure to Directory Distinguished Names[32] by relying heavily on the **Search** operation to locate entries and using

[32] See Section 2.5 for more on Distinguished Names.

the information contained in the Search result as input to **Read** operations if the need arises, without the need for user input. The provision of a tool to enable the creation of such operation templates for the DUA is highly desirable.

3.2.2.6 Informativeness

Surrounding the core functionality of the interface, which enables the user to gain quick and efficient access to Directory information, should be a large catalog of information about the interface itself, it's functionality, the Directory attributes, object classes, and operations that it supports, and so on. Ideally, a naive user will be able to start the package up and obtain meaningful information from the Directory with a minimum of difficulty and without cause to move away from his/her workstation. Some features to look for in this regard are listed below:

- **On-line Help.** Extensive on-line help should be available. The help facility should give clear and concise explanations of any item pointed to on the screen, in addition to offering a writable text window for the entry of help requests. The help facility should have the capability to address topics pertaining to the user interface itself, such that its operation is fully explainable, as well as to address X.500 topics insofar as they pertain to the user interface (for example; attributes, operations, object classes, etc.).
- **Explanation of Errors.** The interface should have the capability to present clear, detailed, plain language explanations of any errors that occur during operation, including errors related

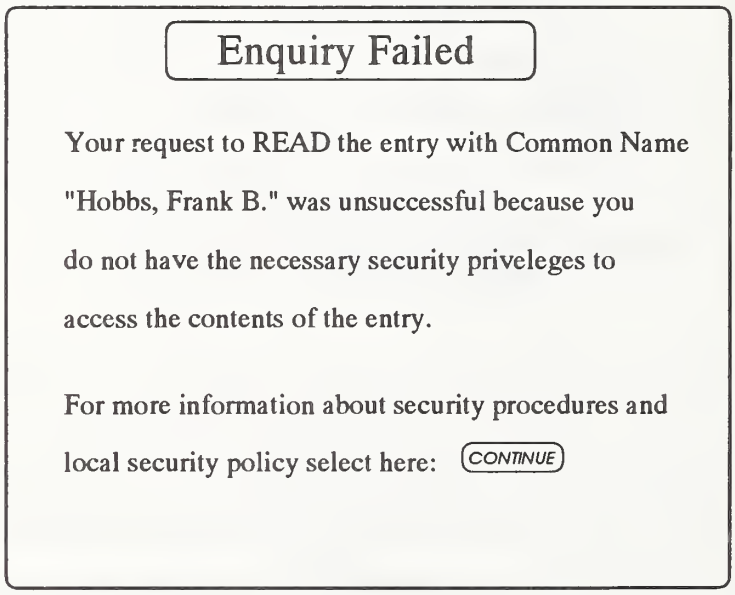


Figure 3.5: *An example of a plain language error notification.*

to the Directory, communications or interface. The provision of multiple-level, branched, menu-driven explanations would be a great asset, providing terse, high-level explanations for expert users as well as detailed explanations for the novice user, and a range in between. Figure 3.5 shows an example of such an error notification.

3.2.3 Directory System Agents

As with DUAs, the range of non-standard features which might be associated with a DSA product is likely to be very large, and we can only hope to give some general pointers as to what the user, or more likely, the system administrator, should investigate.

There are a number of broad subject categories to address when discussing the functionality of DSAs, which may be summarized as follows: *Configurability* of the product ensures that the operator can tailor the operation of the package to best fit the administrative policy for the Directory which has been set up by his/her organization; *Operator Tools* enable the operator to administer the DSA as efficiently as possible; An *SQL* or other interface to a conventional Database Management System (DBMS) means that the DSA package can operate in combination with a regular DBMS package.

3.2.3.1 Configurability

As with DUAs, all aspects of the operation of the DSA should be as fully configurable as possible. For each feature, a clear, easy-to-use tool should exist, again preferably windows-based, to enable the operator/administrator to configure the DSA in accordance with policy requirements quickly and with as few errors or problems as possible, or there should be clear information available as to which files should be edited in order to bring about the desired configuration. Extensive help facilities should also exist (see sec. 3.2.2.6).

A selection of aspects of DSA operation for which configurability is crucial is listed below:

- **Security Configuration.** The X.500 (1994) specifies powerful security features [33] which allow strong authentication using public key cryptosystems and access control to information on various levels, including Administrative Area, Entry, Attribute and Attribute Value. All aspects of an organization's security policy should be easily configurable on each of its DSAs, either through the use of a specialized tool or through simple editing of plain text configuration files.[34]

- **Schema Configuration.** The DSA package should enable full implementation of the organization's X.500 schema, allowing for the definition of non-Standard attribute types, object classes, matching rules, DIT content rules, DIT structure rules and name forms,[35] as well as supporting those defined in the Standard, Reference [1].

3.2.3.2 Operator Tools

Each DSA is a node which holds a part of the global Directory Information Base. As such, it should be regarded as an essential information server and assigned a commensurate number of staff[36] to

[33] see Section 2.8 and Reference [10] for discussions of X.500 Security features.

[34] Since these files contain detailed information on security policy, they should themselves be carefully protected.

[35] See Section 2.5 and Reference [3] for discussions of X.500 Schema elements.

[36] Clearly this number is at the discretion of the owner organization and depends on the scale and operational status of the server in question. A small academic server may require a part time individual from time to time, while a DSA holding a large part of a highly active corporate or governmental DIB may require round-the-clock supervision.

administer security and schema policies, to set up and monitor replication agreements, to load the data initially, making sure it complies with schema and security requirements, to perform backup and restoration of Directory data, and generally to supervise the DSA's operation, to respond to any problems that might arise and provide general user services.

In order to assist the DSA administrator(s) in the efficient performance of these tasks, the DSA package should provide a collection of clearly documented tools enabling the administrator to attend to the tasks with a minimum of effort. Again, it is envisioned that a windows-based interface will be most suitable at this time. Some examples of administrator functions and how they might be carried out are included below:

- **Schema Design.** A large scale X.500 deployment, either within an organization or spanning multiple organizations, will be made up of a number of Directory Management Domains, DIT Domains and Administrative Areas, each with its own policies on schema, security and administration. The DSA operator's toolkit should contain tools to simplify the definition of new schema elements by; for example, allowing the operator to specify an existing schema element from which a new schema element is to be derived and allowing her/him to add or subtract elements, name the new element, modify the inheritance hierarchies to which the element belongs,[37] and so on, in an editable window.

- **Administration and Maintenance.** As with any database, the DSA package requires a set of tools to carry out fundamental maintenance functions, such as backing up and restoring the DIB as necessary, printing out portions of the DIB, reviewing access control policy, cleaning up inactive DIB sections, and so on. However, because the Directory is distributed, other administrative functions are also required, such as the configuration of knowledge references,[38] management of operational bindings (including shadowing agreements)[39] and so on.

- **Bulk Data Load.** When the DIB segment held by a given DSA is initially populated or when the DSA needs to be reloaded from back-up for some reason (equipment failure, security breach or other unforeseen circumstance), a method of loading, the entire DIB segment quickly from a bulk data file may be required. The Standard, Reference [1], does not specify how this should be done, so it is left to the vendor. At the very least, the DSA package should be equipped with some facility which will take bulk data from some sort of file and load it into the DIB in a format suitable for the DSA to read. The tool might also contain functions enabling the operator to selectively load or reload parts of the database, and might present the operator with an easy-to-interpret interface, perhaps giving a schematic of the DIT contained in the back-up file, allowing the operator to select which sections to load by clicking on an entry in the diagram.

  For initial population of the DIB, a Directory Synchronization package will almost certainly be required. Such a device enables the DIB administrator to create a mapping utility to map from any existing Directory facility into a format suitable for the DSA, and thus facilitates the transition from a pre-existing, proprietary directory or database facility to one based on X.500.

  The actual mechanism of getting the data into the DIB may be proprietary/internal, in which case the DIB will be populated from a back-up file on the same system as the DSA in a vendor-specific manner; or the mechanism may use a specialized DUA to populate the DIB using

[37] See Sections 2.4 and 2.3 for discussions of inheritance of characteristics in schema elements.

[38] See Section 2.5 for more information on knowledge references.

[39] See Section 2.10 for information on shadowing agreements.

multiple X.500 **Add Entry** operations.[40] The proprietary method is likely to be quickest, particularly if the DSA and the back-up are collocated on the same machine. The use of a specialized DUA, however, permits the loading of the DSA across a network, and thus the maintenance of a back-up database at a separate location from the Directory System Agent. It should be borne in mind, though, that if the DIB is large, such an operation may have a significant impact on network performance for other network users.

- **Activity Reporting Package.** It is highly desirable for the DSA package to include an activity reporting feature to enable the operator to easily review various usage statistics, such as frequency of contact by other DSAs or by DUAs, frequency of access to various parts of the DIT, security warnings, and so on. The ability to display such information in a readily interpretable form (e.g., graphically) is highly desirable. Access to this information can assist the DSA operator in configuring his/her DSA to run most efficiently (e.g., by maintaining most-accessed information in core memory instead of on disc) and can aid in the management of operational bindings and replication agreements between Directory System Agents.

3.2.3.3 Relational Database Management System Interface

One approach to the database management aspects of DSAs has been to implement the DSA as a front-end package to an existing Database Management System (DBMS) or a Relational DBMS (RDBMS). Typically, an RDBMS using Standard Query Language (SQL) is most suitable for this type of product, since the DSA product can then be run as an application using a variety of RDBMS products supporting the Structured Query Language.[41] Figure 3.6 shows a schematic representation of such an arrangement.

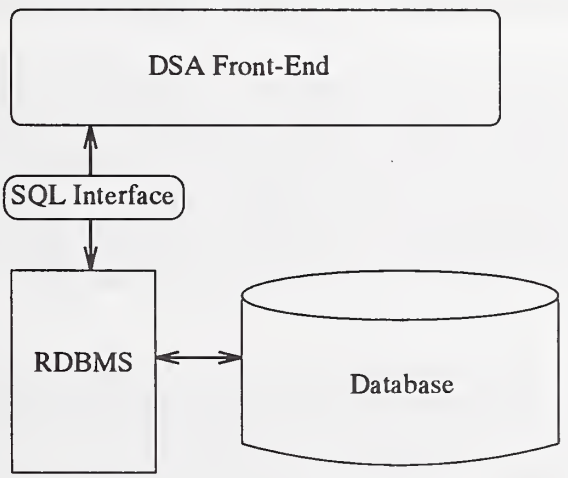


Figure 3.6: *Schematic representation of a DSA front-ending to a conventional Relational Database Management System.*

The DSA front-end adapts the relational structure of the RDBMS to mimic the hierarchical structure of the Directory Information Tree. The operator uses the management tools associated with the RDBMS to manage the DIB, although some activity statistics will still need to be provided by the DSA application (e.g., frequency of contact with other DSAs, security warnings, etc.).

[40] See Section 2.6 for information on the X.500 Add Entry operation.

[41] At least in theory!

Such a product configuration is desirable because it promotes an open architecture, allowing the customer to choose between DSA and DBMS applications which best suit her/his needs. In particular, the customer may purchase a DSA application to match an installed DBMS package without duplicating functionality.

Chapter 4

Performance Evaluation of X.500 Products

This chapter provides guidelines for the evaluation of the performance of an X.500 product. Performance in terms of the Directory is assessed from two perspectives, and a testing methodology for measuring aspects of X.500 product performance is outlined.

4.1 Aspects of Directory Product Performance

Performance in terms of the Directory can be viewed from two perspectives, which might be termed *hard* and *soft*.

The hard aspects of Directory performance are those quantities which can readily be measured, such as the time required by a DSA to process a given operation under controlled or laboratory conditions. These measurements can be used as indicators of how the DSA might perform under real world conditions (though the comparative complexity and unpredictability of a real world situation preclude the use of these measurements as predictors). Measurements of hard performance are couched in terms of tasks completed per unit time, i.e., the unit of measurement is essentially temporal.

The soft aspects of Directory performance refer to more nebulous, less readily measurable properties of the product, which contribute indirectly to the hard performance of any given DSA, or may contribute to the performance of the Directory as a whole, but not necessarily to that of any particular DSA or operation. Such qualities might include the "effectiveness" of a user interface and the "intelligence" of a Directory System Agent.

Note: In the sections that follow, whenever a measurement is specified, the implicit assumption is that the specified operation can be completed (i.e., the target entry exists and contains the necessary information) such that the only subject at issue is the time taken to complete an operation which is known in advance to be possible. Also, these measurements are designed for use under controlled conditions. Comparisons between the products of different vendors thus rely on differences in response times for a predefined operation on a predefined DIT, held in different systems of the same type.

4.1.1 Hard Aspects of Directory Performance

These are aspects of X.500 product performance that can easily be measured in terms of the amount of time required for completion.

4.1.1.1 Directory User Agent User Interfaces

- **Operation Request Processing Time.** For each operation, the time taken between the user's indication that he/she is ready to send a predefined (and syntactically correct) request (usually by hitting the **Return** key or selecting the requisite button in a windowing environment) and the subsequent output of a DAP operation by the Directory User Agent.
- **Error Detection.** The time taken for the DUA to report the presence of a predefined set of errors inserted into an operation request.
- **Operation Response Processing Time.** For each operation, the time taken between the receipt of a predefined operation response over the DAP by the DUA and the display of the encoded information on the user's display device.[42]
- **Operation Error/Referral Processing Time.** For each[43] operation, the time taken between the receipt of an operation error or referral over the DAP by the DUA and the display of the encoded information on the user's display device.
- **Automatic Processing of Referrals.** When a DUA has the capability to automatically process DAP referrals, then for a predefined set of referrals, the time between the receipt of a referral over the DAP and the output of a new operation request over the Directory Access Protocol.
- **Startup Time.** The time taken between the invocation of the DUA program and the availability of the DUA services to the user.

4.1.1.2 Directory User Agent Programmatic Interfaces

Since the DUA Programmatic Interface usually forms an integral part of the DUA User Interface, in that it provides the protocol-related services of the DUA, some of the measurements given for the User Interface apply equally to the Programmatic Interface or Application Programming Interface. The comments related to **Error Detection** and **Automatic Processing of Referrals** also apply to the Application Programming Interface. Two additional measurements of note for the API are the time between invocation of the API and output of a DAP operation, for each DAP operation; and, again for each operation, the time between receipt of a DAP result or error and the presentation of this information to the invoking entity.

[42] In a normal working environment it is not necessarily desirable to display the information immediately as the user's display may already be in use, but for our purposes here the time to display is an appropriate measurement.

[43] Or a representative.

4.1.1.3 Directory System Agents

- **The DAP Resolution of Request.** For each operation, the time between the receipt of a DAP request and the output of a DAP response. In this instance, the DSA is known to hold the target entry of the operation request and the target entry is known to hold the requested information.

- **The DSP Resolution of Request.** For each operation, the time between the receipt of a DSP request and the output of a DSP response. In this instance, the DSA is known to hold the target entry of the operation request and the target entry is known to hold the requested information.

- **The DAP and DSP Production of Referral.** For each operation, the time between the receipt of a DAP or DSP request and the output of a DAP or DSP referral.[44]

- **The DAP and DSP Display of Error.** For each operation, the time between the receipt of a DAP or DSP request and the output of a DAP or DSP error. For each operation, time to display each possible error should be measured. Clearly, this measurement requires the deliberate submission of erroneous operation requests. The standard lists which errors are permissible for each operation (see Reference [4]).

- **The DAP or DSP Chaining of Request.** For each operation, the time between the receipt of an operation request which the DSA is unable to satisfy and the output of a single chained operation request. [45]

- **Chaining of Requests – Search Request Decomposition.** When Search Request Decomposition[46] occurs as part of the processing of a Search operation, two measurements are of interest:

  - The time between the receipt of a Search operation request APDU (DAP or DSP) and the output of the final Search request operation APDU resulting from the request decomposition, and
  - The time between the receipt of the final operation result, error or referral APDU resulting from the requests issued as a result of request decomposition and the output of a result, error or referral APDU to the original requestor.

  The former measures how quickly the DSA can parse a Search request and forward the components to the appropriate DSAs, while the latter measures how quickly the results can be reassembled and forwarded to the original requestor.

- **Structure of Directory Information Tree.** Some DSA products are more sensitive to the structure of the DIT than others. Some perform better with a broad, flat DIT (few levels in the hierarchy, but many entries at each level) while others excel within the confines of a deep, narrow tree (many levels in the hierarchy with comparatively few entries at each level).

[44] This measurement could be split into a set of measurements, depending on the type of knowledge reference used to generate the referral.

[45] This measurement could be split into a set of measurements, depending on the type of reference used to generate the chained request.

[46] I.e., the breaking up of a Search operation request into multiple domain-specific requests and the forwarding of these requests to multiple subordinate Directory System Agents. A Search request is thus propagated down the Directory Information Tree. See Reference [4] and Reference [5] for more details.

A product can be evaluated for performance in this regard by selecting a set of operations and running them against DSAs set up under the two configurations.

- **Performance Under Load Conditions.** The best test of a DSA product under load conditions is to see it run in an operational environment, but in such environments it is impossible to standardize measurements of the performance of one product against another, though the experience of the staff involved with the installation is invaluable. In order to test the performance of a DSA product under load conditions (i.e., where multiple operations are being processed simultaneously) in a standard way, a DUA should be set up to fire out multiple operation requests at a DSA or collection of Directory System Agents. The operation requests may come from a batch file and be pre-encoded, in order to speed delivery. The operations should be tailored towards the configuration being tested. The measure of performance here again is time – the speed with which the DSA configuration resolves the set of simultaneous requests it receives.

- **Start-up Time.** The length of time after which a DSA becomes available following a system failure, re-initialization, maintenance, etc.

4.1.2 Soft Aspects of Directory Performance

These are features or qualities of a Directory product which, while not as easily measurable as the hard aspects described above, may contribute significantly to the speed and efficiency of the Directory and hence to the productivity of its users.

- **Efficiency of User Interface.** The notion of efficiency is difficult to define in terms of a Directory user interface, but if we take a holistic approach, we are able to derive a measurable quantity from the definition. Thus, we define the efficiency of the user interface to be the time between the initial formulation in the user's mind of what information to seek, and the user's comprehension of the result of the corresponding operation.[47] This includes the time taken to bring up and fill out the most appropriate interface screen, as well as the time taken for retries if the results returned indicate that an inappropriate request was sent, and so on.

  Since the user interface is the mediator between the user and the Directory, it follows that the more quickly the user can process an information request through the DUA, the more efficient the user interface is.

- **Intelligent Features.** These are features whose presence is like to enhance the efficiency of information management in the Directory. They are not directly measurable, but their presence or absence may be noted.

  - *Automatic Shadowing.* Where possible, DSAs can automatically set up shadowing agreements if heavy use of a subtree is detected. By shadowing the information, the consumer DSA reduces access time and network congestion.
  - *Automatic Cross-Referencing.* If heavy use of a remote DSA for which no knowledge reference exists is detected, a DSA might automatically add a cross reference to the remote DSA to its knowledge base.

[47] This definition assumes that all other factors, such as DUA operation processing time, DSA operation processing time and the time for the request and result APDUs to travel between the DUA and DSA, are constant. If they are not, then they must be factored out.

- *Caching of an Entry by a Directory User Agent.* If a DUA detects that heavy use is being made of a particular entry, it may automatically issue a Read request, cache the result, and use the cached information to satisfy further queries. In this instance, the benefits in terms of access time need to be weighed against the loss of confidence in the accuracy of the information, since no standard replication agreement exists. This option should be subject to pre-authorization by the user.
- *Automatic Resolution of Referrals.* This is a standard feature which saves the user from the task of re-initiating an operation if it was not possible to chain it within the Directory.
- *Shielding User from Distinguished Names.* To increase the efficiency of the user interface, as discussed above, the user should be shielded from Directory Distinguished Names. This can be done through use of the Search and List operations, showing only the Relative Distinguished Names and allowing the user to select on them in order to initiate other operations. In this way, the Distinguished Names of entries are held purely internally to the user interface.

4.2 Performance Evaluation Methodology

The Performance Evaluation Methodology for X.500 Directory products will be detailed in a future publication which will describe the methodology and report on its practical application in a laboratory environment.

In broad terms, as discussed above, the evaluation of a product will involve the incorporation of timers around the software units comprising the Directory product and the measurement of the time taken to perform standard tasks under controlled conditions.

Bibliography

[1] ITU-T Recommendation X.500 Series (1994) — ISO/IEC 9594,1–9:1994, *Information Technology – Open Systems Interconnection – The Directory*

[2] ITU-T Recommendation X.500 (1994) — ISO/IEC 9594-1:1994, *Information Technology – Open Systems Interconnection – The Directory: Overview of Concepts, Models and Services*

[3] ITU-T Recommendation X.501 (1994) — ISO/IEC 9594-2:1994, *Information Technology – Open Systems Interconnection – The Directory: Models*

[4] ITU-T Recommendation X.511 (1994) — ISO/IEC 9594-3:1994, *Information Technology – Open Systems Interconnection – The Directory: Abstract Service Definition*

[5] ITU-T Recommendation X.518 (1994) — ISO/IEC 9594-4:1994, *Information Technology – Open Systems Interconnection – The Directory: Procedures for Distributed Operation*

[6] ITU-T Recommendation X.519 (1994) — ISO/IEC 9594-5:1994, *Information Technology – Open Systems Interconnection – The Directory: Protocol Specifications*

[7] ITU-T Recommendation X.520 (1994) — ISO/IEC 9594-6:1994, *Information Technology – Open Systems Interconnection – The Directory: Selected Attribute Types*

[8] ITU-T Recommendation X.521 (1994) — ISO/IEC 9594-7:1994, *Information Technology – Open Systems Interconnection – The Directory: Selected Object Classes*

[9] ITU-T Recommendation X.509 (1994) — ISO/IEC 9594-8:1994, *Information Technology – Open Systems Interconnection – The Directory: Authentication Framework*

[10] ITU-T Recommendation X.525 (1994) — ISO/IEC 9594-9:1994, *Information Technology – Open Systems Interconnection – The Directory: Replication*

[11] *IGOSS-Industry/Government Open Systems Specification*, National Institute of Standards and Technology Special Publication 500-217, May 1994.

[12] *Stable Implementation Agreements for Open Systems Interconnection Protocols, Version 7, Edition 1, December 1993*, National Institute of Standards and Technology Special Publication 500-214.

[13] *IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Part 17: Directory Services Application Program Interface (API)*, Institute of Electrical and Electronics Engineers Standard 1003.17.

[14] OSINET X.500 Directory Services Stable Interoperability Tests, Version 1.0, Edition 1.0, June 8, 1993.[48]

[15] Rose, Marshall. T. and Cass, Dwight, E., *ISO Transport Services on top of the TCP*. Request for Comments 1006, DDN Network Information Center, SRI International, May, 1987.

[48] This document is available through the Corporation for Open Systems, 8260 Willow Oaks Corporate Drive, Suite 700, Fairfax, VA 22031

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SYSTEMS TECHNOLOGY

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500–.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published bimonthly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program in support of the efforts of private-sector standardizing organizations.

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and nongovernment). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Institute of Standards and Technology
Gaithersburg, MD 20899