Deployment Guide

Netscape Directory Server

Version 4.1


Recycled and Recyclable Paper

Version 4.1

01 00 99 10 9 8 7 6 5 4 3 2 1

Netscape Communications Corporation 501 East Middlefield Road, Mountain View, CA 94043

Contents

Introduction 9
What Is in This Book? 9
Conventions Used in This Book 10

Chapter 1 Welcome to the Directory Server 11
About the Enterprise 12
What Is a Directory Service? 12
N+1 Directory Services 13
Global Directory Services 14
LDAP 15
The Netscape Directory Server 15
Client-Server Architecture 15
Server-Side Architecture 16
Directory Server Concepts 17
The Directory Tree 17
Distinguished Names 18
Suffix 18
Root Entry 19
Directory Manager 20
Base Distinguished Name 21
Secure Sockets Layer 21

Chapter 2 Directory Deployment Overview 23
Directory Design Activities Overview 24
Deployment Advice 25

Chapter 3 Planning Your Directory Data 27
Data Planning Overview 28
Introduction to Directory Data 29
Examples of Directory Data 30

What Your Directory Should Not Include ... 30
Data Planning ... 31
Performing the Site Survey ... 33
Analyzing Your Site Survey ... 34
Data Mastering ... 34
Data Mastering for Replication ... 35
Data Mastering Across Multiple Applications ... 35
Data Ownership ... 36
Determining Data Access ... 37
Documenting Your Site Survey ... 39
Directory Schema Design ... 40

Chapter 4 Planning Directory Schema ... 41
Directory Data Representation Overview ... 42
Object Classes ... 43
Standard Object Classes ... 43
Inheritance ... 43
Attributes ... 44
Required Versus Optional Attributes ... 45
A Note About Attribute Values ... 45
Schema Checking ... 47
Customizing the Schema ... 47
When You Should Extend Your Schema ... 47
When to Add New Object Classes and Attributes ... 48
Object Class Strategies ... 48
A Consistent Schema ... 50
Schema Extension Frequently Asked Questions ... 52

Chapter 5 Planning Security Policies ... 55
Security Policy Overview ... 56
Directory Access Rules ... 57
User Authentication ... 57
Certificate-Based Authentication ... 58
Authenticating Using Directory Server for NT ... 59
Anonymous Access ... 60

Directory Manager Authentication 61
Access Control Lists 61
Setting Permissions 62
The Precedence Rule 62
Allowing or Denying Access 63
When to Explicitly Deny Access 63
Where to Place Access Control Rules 64
Using Filtered Access Control Rules 65
The ACI Format 66
Targets 67
Permissions 67
Bind Rules 68
Using ACIs: Some Hints and Tricks 69
Creating a Security Policy 71
Planning Your Groups 71
Group Usage Advice 73
Determining General Directory Access 73
Determining Points of Access 74

Chapter 6 Directory Tree Design 77
Directory Tree Overview 78
Multiple Suffixes 79
Planning Your Directory Suffix(es) 80
Selecting a Suffix 80
Using Multiple Suffixes: ISPs 81
Using Multiple Suffixes: Large Enterprises and Extranets 82
Branching Your Directory Tree 84
Branch Point Distinguished Name Attributes 84
Branching Strategies 86
Branching to Support Replication or Referrals 88
Branching to Support Access-Control 89
Branching to Support International Enterprises 90

Naming Person Entries ... 91
Naming Non-Person Entries ... 93

Chapter 7 Planning Replication ... 95
Replication Overview ... 96
Supplier Servers ... 97
Consumer Servers ... 97
Replicating Directory Trees ... 98
Cascading Replicas ... 99
Subtree Replication ... 100
Multiple Subtree Replication ... 100
Initiating Replication ... 102
Directory Entries used to Support Replication ... 103
Supplier-Initiated Replication ... 103
Consumer-Initiated Replication ... 104
Building a Highly Available Directory Service ... 104
Using Replication for High Availability ... 105
Using Replication for Load Balancing ... 107
Load Balancing the Server ... 108
Load Balancing the Network ... 108
Using Replication for Local Availability ... 109
Determining Your Replication Strategy ... 110
Example: Load Balancing a Small Site ... 112
Example: Load Balancing a Large Site ... 112
Example: Load Balancing for Local Management ... 113
Example: Load Balancing for Server Traffic ... 114
Example: Replicating for Messaging 3.0 ... 116

Chapter 8 Planning Referrals ... 119
Referral Overview ... 120
Smart Referrals ... 120
Smart Referral Usages ... 123
How You Should Use Smart Referrals ... 123
LDAP Client Referral Handling ... 125
Netscape Client Referral Handling ... 125

Chapter 9 Directory Design Examples 127
A Small Organization 128
A State Government 131
An International Corporation 133
Single Suffix, Global Directory Tree Replication 133
Single Suffix, Local Data Management 135
Multiple Suffixes, Local Data Management 136
Enabling the Extranet 139

Chapter 10 Extending Your Directory Service 141
Building Custom Tools for Data Migration 142
Building Custom LDAP Clients 143
The LDAP C and Java SDKs 144
Writing Server Plug-Ins 145

Appendix A Quick Start 147
A Word of Advice 148
Planning Your Suffix Value 148
Directory Tree Advice 149
DN Advice 149
Configuration Directory Advice 150
Creating Directory Entries 151
For More Information 152

Index 153

Introduction

Welcome to the Netscape Directory Server and the Internet. Netscape Communications Corporation is the premier provider of open software that lets people and companies exchange information and conduct commerce over enterprise networks and the Internet.

What Is in This Book?

This manual provides you with a foundation for planning your directory service. It describes how you can use the Directory Server to support simple usage that involves only a few hundred users and some key Netscape server applications. It also shows you how you can scale the Directory Server to support millions of users around the world. Along the way, you are introduced to the general issues and concerns that you should be aware of when deploying a global directory service. You are also introduced to the basic directory service concepts and specific guidelines that you will need to deploy a production-grade directory service.

Note It is assumed that you have already read *Managing Servers with Netscape Console*. That manual provides introductory information on the Netscape administration services.

Because this manual assumes that you are interested in designing a global directory service for your enterprise, you may find that much of this manual's contents assume a level of complexity that is not true of your environment. This is especially likely if you are deploying the Directory Server to support a few Netscape server instances. Further, you may be installing the Directory Server for review or testing purposes or to support an LDAP client development effort.

For simpler deployment scenarios, you can skip most of this book (although it is recommended that you acquaint yourself with the topics discussed here), and instead read just the following:

- Chapter 1, “Welcome to the Directory Server”
- Appendix A, “Quick Start”

For information on how to install a Netscape Directory Server, see the *Netscape Directory Server Installation Guide*.

Conventions Used in This Book

This section explains the conventions used in this book.

`Monospaced font`—This typeface is used for any text that appears on the computer screen or text that you should type. It is also used for filenames, functions, and examples.

Note Notes and Warnings mark important information. Make sure you read the information before continuing with a task.

|—The vertical bar is used as a separator for user interface elements. For example, Configuration|Logs means you should go to the Configuration Tab on the Directory Server Console and then select the Logs icon.

Throughout this book you will see path references of the form:

```
<NSHOME>/slapd-<serverID>/...
```

In these situations, `<NSHOME>` represents the directory where you installed the server, and `<serverID>` represents the server identifier you gave the server when you installed it. For example, if you installed your server in `/export/ns-home`, and gave the server an identifier of `phonebook`, then the actual path would be:

```
/export/ns-home/slapd-phonebook/. . .
```

Also, all paths specified in this manual are in Unix format. If you are using a Windows NT-based Directory Server, you should assume the NT equivalent filepaths whenever Unix file paths are shown in this book.

Chapter

1

Welcome to the Directory Server

The Netscape Directory Server provides the centralized directory service upon which your intranet or extranet is based. Your Netscape servers and other directory-enabled applications use the directory service as a common, network-accessible location for storing shared data such as user and group identification, server identification, and access control information. In addition, you can extend the Netscape Directory Server to support your entire enterprise with a global directory service that provides you with centralized management of all your enterprise's resource information.

In this chapter you will learn about the basic Directory Server concepts that you need to use the Directory Server, regardless of the type of installation that you are planning. This chapter includes the following sections:

"About the Enterprise" on page 12—This section describes the concept of the enterprise, especially as the term is used in this guide.

"What Is a Directory Service?" on page 12—This section explains what a directory service is, what it is used for, and the problem set that a directory service is intended to solve. This section also introduces the Lightweight Directory Access Protocol (LDAP).

"The Netscape Directory Server" on page 15—This section describes the Netscape Directory Server and high-level server architecture.

“Directory Server Concepts” on page 17—This section defines the basic terms and concepts that you need to know to install and use a Netscape Directory Server. Concepts include the directory tree, distinguished names, suffix, root entry, root distinguished name, base distinguished name, and secure sockets layer.

About the Enterprise

The *enterprise* is often defined to mean all aspects of your computing environment -- from the mainframe (if applicable) to the server (Unix or NT) to the desktop level.

However, from the perspective of Directory Server deployment, the enterprise also comprises your organization’s resources and activities. You can understand resources to mean the people and groups within your organization as well as the devices (printers, servers, and desktop systems) that your organization uses, and the applications that you run to support your organization (such as a web server or an accounting package). Further, the enterprise also encompasses the physical location(s) in which your organization resides. These differing locations can be separate buildings, cities, states or provinces, or even countries.

You should therefore consider the term the *enterprise* to mean your entire organization from the buildings that you use, to the software that you run, to the people that you work with.

What Is a Directory Service?

A *directory service* is a collection of software that you use to store information about your enterprise. Usually, but not always, directory services are based on a client-server architecture. Therefore, a directory service generally consists of at least one Directory Server and one or more directory clients.

The basic function of a directory service is to allow you to store information about your enterprise so that it later can be retrieved either by directly searching for that information or by searching for related but more easily remembered information, such as a name.

One fairly well-known client-server directory service is a Domain Name System (DNS) server. Among other things, a DNS server maps computer host names to IP addresses. All of the computing resources on your network are then clients of the DNS server. This allows users of your computing resources to easily locate computers on your network by remembering host names rather than IP addresses, which are a seemingly arbitrary series of numbers to most users.

Of course, a service that stores only two types of information, names and IP addresses, is a limited example of a directory service. A true directory service is used to store virtually unlimited types of information. For example, a true directory service can be thought of as an electronic telephone book in that it almost always contains personal contact information such as a person's name, telephone number, mail address, office number, and so forth. But a Directory Server goes beyond this usage by allowing the enterprise to store other types of information such as:

- physical device information (For example, you can store information about all the printers in your organization, where they reside, whether they are color or black and white, their manufacturer, date of purchase, serial number, and so forth.)
- private employment information such as salary, government identification numbers, home addresses and phone numbers, pay grade, and so forth
- contracts or accounts information, such as the name of the client, final delivery date, bidding information, contract number, and milestone due dates

Ultimately a robust, scalable directory service allows you to store all of your enterprise's information in a single, network-accessible repository.

N+1 Directory Services

Multiple databases serving common directory needs is known as the *n+1 directory problem*, and it has long been a logistical and financial problem, as well as a security hazard for the enterprise.

Until recently, the basic problem has not been finding and deploying directory services, but rather that enterprises are using too many directory services at once. Most of these directories are bundled as proprietary databases within an application. For example, email systems are traditionally heavy users of directory services. As the many different types of proprietary email systems

have spread throughout the enterprise, so too have their accompanying proprietary directory services. These directory services all stored the same types of information: user names, mail addresses, host information, mailbox information, passwords, and so forth.

The problem is that these different directories (some of which were proprietary, some of which were built around standards-based directory services) can not readily share data between themselves. Thus if a user had a mailbox in two different email systems, that user's information usually had to be managed in two different directories. The result is increased costs in both personnel as well as hardware because user information is managed in multiple locations.

Global Directory Services

The concept of a global directory service was invented to solve the n+1 directory problem. The idea was to provide a single, centralized repository of directory information that any application can access. However, the implementation of a global directory service faced some important difficulties. The most serious of these was the question of how the many disparate applications used in the enterprise were supposed to access the directory. The directory would have to be network-based, but what communications protocol would be used for that access?

There were several contenders for the protocol. One of these was the X.500 ISO standard, which is a protocol in use in many enterprises today. X.500 has several advantages that make it attractive for solving the n+1 directory problem. It offers an open communications protocol called the Directory Access Protocol (DAP) that any application can use to access the directory. It also offers an extensible information framework that allows the directory to store virtually any kind of information. Finally, X.500 offers a remarkably robust directory solution that could be scaled to millions of users.

Unfortunately, X.500 also has several limitations. Among these is a dependency on a communications layer that is not the Internet standard TCP/IP protocol, and complicated requirements for directory-naming conventions. The result is a directory strategy that offers the scalability and robustness required by a global directory service but which suffers from expensive administrative costs.

LDAP

LDAP, or the *Lightweight Directory Access Protocol*, was invented to preserve the best qualities offered by X.500 while reducing the administrative costs. LDAP provides an open directory access protocol running over TCP/IP. It retains the X.500 data model and it is scalable to a global size and millions of entries for a modest investment in hardware and network infrastructure. The result is a global directory solution that is affordable enough to be used by small organizations, but which also can be scaled to support the largest of enterprises.

The Netscape Directory Server

The Netscape Directory Server 4.x supports LDAP versions 2 and 3. Each Directory Server instance is capable of supporting millions of entries and thousands of queries per second. Further, through the use of replication and referrals, the Directory Server can be scaled to support even the largest of enterprises, up to and including the multinational corporation.

Without modification, the Directory Server can provide the foundation for your intranet or extranet. Every Netscape server uses the directory as the storage location for shared server information. It is the Directory Server that enables key capabilities such as single-user logon, centralized user and group management, and location independence for your users through its function as a network registry.

Further, you can use the Directory Server to enable a true global directory service that will lower your administrative costs to doing business.

Client-Server Architecture

LDAP directory services are implemented using a client-server architecture. Thus, an LDAP directory service implementation consists of at least one LDAP server and at least one LDAP client. Most directory services consist of multiple LDAP clients communicating to a Directory Server over the network. As directory services grow to include larger numbers of entries or larger numbers of clients spread out geographically, they also include multiple Directory Servers placed in strategic locations around the network.

The Netscape Directory Server product provides the software necessary for an entire directory solution. It includes the Directory Server, which is the server-side software that implements the LDAP protocol, and an HTML interface for end users to search and modify the directory. Other LDAP clients are also available. These include the directory managers in the Netscape Console, and the addressbook feature in Netscape Communicator 4.x. In addition, other LDAP clients are available commercially, and you can write your own LDAP client using the LDAP client SDK that is included with the Netscape Directory Server product. (For an introduction to the LDAP client SDK, see Chapter 10, "Extending Your Directory Service.")

Server-Side Architecture

The Directory Server is essentially a communications engine that stores information in one or more databases. The server is split into two major parts: a front end that is responsible for general network communications and one or more back-end plug-ins that are responsible for database management.

The Directory Server is a multithreaded application that is capable of processing hundreds of thousands of search requests per hour (actual performance is determined by the supporting hardware and network connections). The Directory Server ensures high-performance for search requests by using a database tuned for searching and by building indexes that allow for rapid directory lookups. To further improve performance, the Directory Server caches the most heavily accessed data in memory.

By default, the Directory Server uses a single database plug-in. This database is robust implementation that is capable of managing millions of entries. It supports advanced backup and restore activities that ensure your directory data is always in a consistent state, even in the event of a power outage when data is being written to the database.

For most directory implementations, this default database will support your directory service without any need for customization. However, under some circumstances you may choose to use multiple databases. You can also use the Netscape Directory Server plug-in API to replace this standard database with something else (such as a relational database). This is a non-trivial task that requires a fair amount of development and planning in its own right.

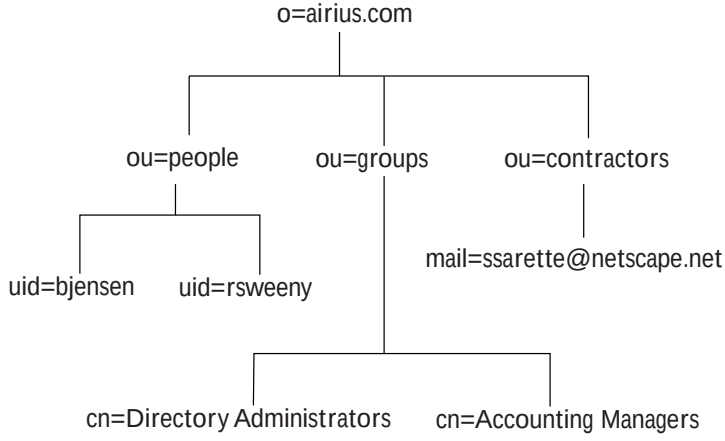
The usage of multiple databases and custom back ends is beyond the scope of this manual. For more information on database usage with your Directory Server, see the *Netscape Directory Server Programmer's Guide*.

Directory Server Concepts

You need to understand a few key concepts before you can deploy an LDAP directory service. Understanding these concepts will save you a great deal of time when you install your Netscape Directory Server.

The Directory Tree

The entries in an LDAP directory service are often visualized as being organized in a tree-like structure. This mirrors the tree model used by most file systems, with the tree's root point (first entry) appearing at the top of the hierarchy. This is referred to as the *directory tree*:



A similar term, *Directory Information Tree* (DIT), is often used to refer to an enterprise's directory tree. However, DIT is strictly defined to mean the global X.500 directory tree. In the X.500 view, there is only one DIT and all directory services are considered to be managing portions of that single DIT.

The concept of a single, global DIT is not as important to LDAP-based directory services as it is to X.500 directory services. Therefore, to avoid confusion with this X.500 term (which many directory architects are familiar with), the Netscape Directory Server documentation simply refers to the *directory tree*, which is meant to be all or part of the directory tree in use in your enterprise.

Distinguished Names

Just as a file path uniquely identifies a file within a file system, a directory entry is uniquely identified within the directory tree using a distinguished name (DN). A DN identifies the entry by using a series of comma-separated attributes and attribute values. However, the path specification for DNs is in reverse order from a traditional file system. That is, where a file system typically traces the path to the file from left to right with the file's actual name specified in the right-most component, a DN specifies the left-most component as being the actual directory object and the right-most value as being the directory root point.

Thus, a DN might be

```
uid=bjensen, ou=people, o=airius.com
```

This DN represents the entry named *bjensen* in the subdirectory named *people* in the directory named *airius.com*.

The DN's left-most value is known as the Relative Distinguished Name (RDN). Following this value are subsequent attributes that represent a branch point above the entry. The final, or right-most, attribute represents the conceptual root point of the directory tree.

Suffix

A Directory Server's suffix value identifies the directory tree maintained by the server. That is, a suffix is always equal to the directory tree's root entry (see the example below). Suffixes are represented in DN format. Every Netscape Directory Server has multiple suffixes defined for it. Only one, however, is of general importance, and this is called the *primary suffix*. The primary suffix represents the directory tree under which you are storing directory information

of interest to your enterprise. You can have more than one primary suffix defined for your server. That is, you can have more than one directory tree managed by your server.

For example, a Directory Server that contains an entry named

```
uid=bjensen, ou=people, o=airius.com
```

might have a suffix named

```
o=airius.com
```

However, suffixes do not have to be limited to a single component. In fact, it is very common for a suffix to contain at least two components. For example, the entry

```
uid=bjensen, ou=people, dc=airius, dc=com
```

might have a suffix named

```
dc=airius, dc=com
```

Your server will always have other, secondary suffixes defined by it that are used internally by the Directory Server. These include a suffix that represents your server's machine data area, a suffix used for server administration (`o=NetscapeRoot`), and other suffixes used for server housekeeping such as schema management and server configuration management. For details on Directory Server suffixes, see the *Netscape Directory Server Administrator's Guide.*

Root Entry

The root entry is the first, or top-most, entry in your directory tree. The root entry must have a DN that is identical to a suffix defined for your directory. Thus, if you have the suffix `o=airius.com` then the root entry (that is, the first entry) in your directory tree must have the distinguished name of `o=airius.com`

Similarly, if the suffix contains multiple attributes (`dc=airius, dc=com`), then the DN of the first entry in the directory tree must be `dc=airius, dc=com`.

Directory Manager

The Directory Manager is the special entry to which access control does not apply. Think of the Directory Manager as your directory's superuser. In previous versions of the Directory Server, the Directory Manager was known as the root DN, and this term is still used in the Directory Server's configuration file, as well as in some of it's user interface screens.

By default, the Directory Manager uses no suffix, and it is simply a common name attribute-data pair:

```
cn=Directory Manager
```

The Directory Manager is not required to have a corresponding entry in the directory. As a result, if you search your Directory Server for your Directory Manager entry, you usually will not find it (unless you explicitly created a corresponding entry for the Directory Manager). Also, unless you intend to create an actual directory entry that corresponds to your Directory Manager, your Directory Manager does not have to correspond to any of the suffixes managed by your server.

All information regarding the Directory Manager and its password is stored in the Directory Server's configuration file (`slapd.conf`). The Directory Manager's password can be stored in clear text or in encrypted form (by default it is encrypted).

Configuration of a Directory Manager for your server is optional, but it is strongly recommended because without it the creation of directory entries and the initial setting of access control privileges is difficult. You can configure your Directory Manager when you first install your Directory Server instance. You can also create a Directory Manager, change your Directory Manager, or delete your Directory Manager after you have installed your server. For more information on managing the Directory Manager, see the *Netscape Directory Server Administrator's Guide*.

Base Distinguished Name

The *Base Distinguished Name*, or base DN, is the entry in your directory from which searches will occur. This search point is determined by settings created on each individual LDAP client using your directory.

The base DN is also often referred to as the *search base*. In addition, Netscape Communicator's preference settings for directories refers to this as the *search root*.

For example, if you specify a base DN of `ou=people, o=airius.com`, then the search operation will examine only the `ou=people` subtree in the `o=airius.com` directory tree.

Every LDAP search request must specify a base DN. Most often you will configure your LDAP client such that it uses your directory's root entry for the base DN. In this way, your LDAP client will always search the entire directory tree when performing searches.

Secure Sockets Layer

You can set up your Directory Server to use encryption for server communications. Encryption is most important to directory services running over public networks such as the Internet.

Directory server encryption is performed using the Secure Sockets Layer, or SSL. Encrypted LDAP communications are referred to as *LDAPS connections*.

SSL is used throughout the Netscape Server Family to perform other security activities such as message integrity checks, digital signatures, and mutual authentication between servers as well as between servers and clients.

In order for your server to use LDAPS, you must configure a security database for the server, and then turn on SSL in the server. Your server is capable of simultaneously handling both LDAP and LDAPS communications.

For more information on SSL, see *Managing Servers with Netscape Console*. For more information on LDAPS, see the *Netscape Directory Server Administrator's Guide*.

Chapter

2

Directory Deployment Overview

This manual provides information and advice on how to design a robust, secure directory service that is both easy to scale and easier still to manage. As part of this design goal, this manual helps you avoid unnecessary complexity in your directory design.

As you go about designing your directory service you should follow the general philosophy that a simple design is a better design. While it is not possible to entirely avoid complexity, you should strive for simplicity in all aspects of your directory design.

In this chapter you will learn about the directory design process. This chapter includes the following sections:

"Directory Design Activities Overview" on page 24—This section briefly describes the various planning activities that you must perform to successfully design and roll-out a directory service.

"Deployment Advice" on page 25—This section provides high-level advice on physically installing your directory service into your production environment.

Directory Design Activities Overview

The most important task to ensuring a successful directory service is planning. As with all software deployments, good planning in the early stages of the deployment will save you time and money as your directory service grows and matures. You should therefore budget plenty of time to plan your directory's contents, directory access-controls, physical placement of Directory Servers, and replication strategies.

However, remember that the Netscape Directory Server is very flexible. It is possible for you to easily rework your directory service after the initial deployment phase to meet unexpected or changing requirements. Therefore, your goals in planning your service should be to anticipate your enterprise's needs as much as possible, but you should not feel that you have to know every single detail in advance.

There are a few major areas to focus on when planning your directory service. Each of these areas have some overlap with other areas, but if you plan for each of these aspects of your directory service, you will have a good understanding of your directory's requirements and design.

Because these areas are heavily interrelated, the order in which you approach them is up to you. However, you may find that it is useful to examine each of these areas in the following order:

- Plan your directory contents. This includes determining what data you want to keep in your directory and how you will represent that data in the directory. As a part of this planning, you must also locate the data in your enterprise. It is very likely that you will find that the information that you want to keep in your directory is currently contained in many different locations, and potentially mastered by multiple organizations. You may also find that some of the data you want to keep in your directory is not currently maintained in your enterprise and so processes will have to be developed to maintain this information.

- Plan data management. Who will own (add, modify, or delete) each piece of information and who can read it? Also, determine the physical location where the master of each piece of data will be kept.

- Plan access-control. This is related to data management. Who can access what data in your directory? Who can read the data? Who can write to the data?

- Plan groups. This is also related to access-control and data management. Decide what groups you will maintain in your directory, and what special privileges, if any, will you grant each group.
- Plan replication. This entails deciding where you are going to physically place Directory Servers, and which servers will be responsible for mastering which data. It is not required, for example, to place all of your enterprise's data on every server. Instead, you may want to replicate just the information to a server that is of interest to the people or applications near that server.
- Plan referrals. This is directly related to replication management. If a server will not contain information of general use to your enterprise, then you may want to create smart referrals to a server that does contain the data.
- Plan the directory tree. After you have spent some time examining your enterprise's directory requirements you will be in a much better position to understand what your directory tree will look like. Expect the tree's layout to change multiple times as you come to a better understanding of your enterprise's requirements. Because directory tree design is often fluid in the early stages of directory planning, it is best to sketch the design on paper and not attempt to represent it on an actual Directory Server until the other aspects of your design have been finalized.

Deployment Advice

After you have planned your directory service, you can start to physically install your directory service into your production environment. As is the case with directory design, the most important aspect of your directory deployment is planning. By spending the time to plan the installation and population of your service, you will increase your users' confidence in the directory by avoiding unexpected and embarrassing outages.

To help you keep your deployment problem set to a manageable size, you should use a phased deployment that implements well-defined aspects of your directory service in stages. This will in turn help to reduce the total time and head count needed to deploy your directory service, and thus reduce the total cost of ownership for the directory.

Finally, by using a well-planned, phased deployment strategy, you can locate unexpected problems in your design before they become mission-critical.

Keep this deployment advice in mind:

- Start small. Instead of simultaneously installing every Directory Server that will be a part of your directory service, install just a few at first. Populate those few servers with the immediately necessary information and then provide some well-defined group of users access to the service. See how your users actually use the service (what their access habits are like) and how well your network infrastructure responds to the increased network load. Be prepared to add or subtract servers as usage patterns become clear.

  Because you are centralizing information, your directory service must be available at all times. By gradually deploying your directory service, you can ensure that the supporting hardware, networks, and software are up to handling the increased loads.

- Grow over time. Once you have experience with how your directory service is performing for the initial roll-out, continue to deploy in stages. Do not try to install your entire directory all at once, especially if you are deploying into a large, complex environment. Instead, continue extending the service to other core locations around your enterprise in stages. Do this not only with the installation and configuration of your Directory Servers and clients, but also with the migration of data from legacy directories to the LDAP directory.

- Deploy in parallel with existing directories. Give your new directory service some time in your production environment before dismantling your legacy directories and databases (if that is your goal). Start with the LDAP directory running as a secondary or backup service to your existing infrastructure. Once the LDAP directory has proven itself, you can start to use it in the primary role and downgrade your legacy systems to the secondary or backup role. After a sufficient amount of time has passed without any significant problems, you can eliminate your legacy directories altogether, if you so desire.

  The amount of time that should pass before you rely entirely on your LDAP directory depends on the state of your computing environment, including the reliability of your networks, the complexity of your data management tasks, and the availability of LDAP-enabled software that allows you to manage information directly in the LDAP directory. You will find that the biggest hurdle to moving to an LDAP-only environment is getting all your mission-critical applications to use LDAP.

Chapter

3

Planning Your Directory Data

Your directory data is the information that you contain in your directory service. This data will include common information such as users' names, contact information (such as email addresses and telephone numbers), group identification, and group membership. A large part of designing your directory service is planning your directory's content.

In this chapter you will learn about the issues and strategies behind planning your directory's content. This chapter includes the following sections:

"Data Planning Overview" on page 28—This section provides an overview to the planning activities that you will perform while planning your directory's contents.

"Introduction to Directory Data" on page 29—This section describes what should and should not be included in your directory. Examples of the kind of data that is a good candidate for your directory is provided, as well as the kind of data that you should avoid placing in your directory.

"Data Planning" on page 31—This section provides advice on how to approach your data planning tasks.

"Performing the Site Survey" on page 33—This section provides advice on surveying your site for directory data.

“Analyzing Your Site Survey” on page 34—This section tells you how to approach data management in your directory. The concepts of data mastering, data ownership, and data access are discussed. Finally, some advice is given as to how you can document the results of your site survey and data analysis.

Data Planning Overview

Planning your directory's data is the most important aspect of your directory planning activities. Therefore, you should budget plenty of time for data planning.

You will spend the majority of your time surveying your enterprise to locate all the data stores where directory information is managed. As you perform this survey, expect to find that some kinds of data are not well managed; some processes may be inefficient, inadequate, or nonexistent altogether; and some kinds of data that you expect to find are not available at all. All of these issues should be addressed before you finish your data-planning phase.

Your data-planning activities should include:

- Determine what directory-enabled applications you want to deploy and what their data needs are.
- Survey your enterprise and identify where the data comes from (such as NT or Netware directories, PBX systems, Human Resources databases, email systems, and so forth).
- Determine who needs access to the data. In particular, pay attention to your enterprise's mission-critical applications. Find out if those applications can directly access and/or update the directory.
- For each piece of data, determine the location where it will be mastered.
- For each piece of data, determine who owns the data; that is, who is responsible for ensuring that the data is up-to-date.
- For each piece of data, determine the name of the attribute that you will use to represent the data in the directory and the object class (the type of entry) that the data will be stored on.

- If you are going to import data from other sources, develop a strategy for both bulk imports and incremental updates. As a part of this strategy, try to master any given piece of data in just a single location, and limit the number of applications that can change the data to as few as possible. Also, keep the number of people who can write to any given piece of data to a small, easily identifiable group. Doing this will help ensure your data's integrity while greatly reducing your enterprise's administrative overhead.

  Remember that simpler is better when it comes to managing data sources.

- Document your findings.

The following sections describe the data-planning activities in detail.

Introduction to Directory Data

The nature of the data that you contain in your directory is up to you, however some types of data are better suited to a directory service than others. Ideal candidates for inclusion in a directory service have some subset of the following characteristics:

- The data should typically be read much more often than it is written. This is because directory services are tuned for read operations; write operations are considerably more expensive than reads in that they slow your server's performance down with respect to its intended usage.

- The data must be expressible in attribute-data format (for example, `surname=jensen`).

- The data should be of interest to more than one audience. For example, an employee's name or the physical location of a printer can be of interest to many people and applications.

- It should be useful to access the data from more than one physical location. For example, an employee's preference settings for a software application may not be a candidate for inclusion in the directory because only a single instance of that application needs access to the information. However, if the application is capable of reading preferences from the directory, then it is very useful to include the preference information in the directory. Doing so allows the user to interact with the application according to her preferences, regardless of where the user is physically located within the enterprise.

Examples of Directory Data

The following are typical examples of directory data:

- a person's contact information, such as telephone numbers, physical addresses, and email addresses
- a person's descriptive information, such as an employee number, job title, manager or administrator identification, and job-related interests
- an organization's contact information, such as a telephone number, physical address, administrator identification, and business description
- device information such as a printer's physical location, type of printer, whether the printer is capable of color output, and the number of pages per minute that the printer can produce
- contact and billing information for your corporation's trading partners, clients, and customers
- contract information, such as the customer's name, due dates, job description, pricing information, and general contact information for both the customer as well as the personnel within your enterprise responsible for the contract
- a person's software preferences or software configuration information
- resource locations, such as pointers to web servers or the file system location of a certain file or application

What Your Directory Should Not Include

A directory service is not a file system, a file server, an ftp server, a web server, or a relational database. Therefore, if you want to include large, unstructured objects in your directory, you should consider using a server more appropriate for the task. However, it is appropriate to store pointers to these kinds of applications within your directory service through the use of FTP, HTTP, or other types of URLs.

Remember that a directory service is not a replacement for a relational database, although you can use a relational database to store directory data (see **“The Netscape Directory Server” on page 15** for details). Therefore, you should avoid placing any data that needs a relational data mode into your directory.

Also, because the directory is tuned for read operations, you should avoid placing rapidly changing information in the directory. Reducing the number of write operations occurring in your directory service maximizes overall search performance.

Data Planning

Generally data planning should be driven by the applications that access your directory and the data needs of these applications. Some of the more common applications that you will use with your directory include:

- A directory browser application, such as an online telephone book. Decide what information (such as email addresses, telephone numbers, employee name, and so forth) you want your users to be able to obtain through the directory when doing telephone book lookups and make sure you put that kind of information into the directory.
- Email applications, especially email servers. Not all email servers will require the same types of information. All email servers require email addresses, user names, and some routing information to be available in the directory. Others, however, will require more advanced information such as the location on disk where a user's mailbox is stored, vacation notification information, and protocol information (IMAP versus POP, for example).
- Directory-enabled HR applications. These require more personal information such as government identification numbers, home addresses, home telephone numbers, birth dates, salary, and job title.

When you are planning your directory data, plan not only what you want to place in your directory today, but also try to determine what you want to include in the directory at some point in the future. While not strictly necessary, planning ahead can help you scale your directory service to take on bigger roles in your enterprise.

As you plan, consider these points:

- What do you want to put in your directory today? That is, what is your immediate problem that you hope to solve by deploying a directory service? What is immediately needed by the directory-enabled applications that you will use first?
- What do you want to put in your directory in the future? For example, your enterprise might use an accounting package that does not currently support LDAP, but which you know will be LDAP-enabled in the near future. You should identify the data use by applications such as this and plan for the migration of the data into the directory when the technology becomes available.
- What do you think you might want to someday store in your directory? While this is the hardest case of all to consider, doing so may pay off in unexpected ways. At a minimum, this kind of planning helps you identify data stores (that is, locations where information is managed) that you might not otherwise become aware of.

If you are going to use your Directory Server for more than just Netscape server administration, then you will have to plan the type of information that you will store in your directory. Looking beyond Netscape servers, you may find that you want to include information such as:

- contracts or client accounts
- payroll
- physical devices
- home contact information
- office contact information for the various sites within your enterprise

Performing the Site Survey

To identify all of the data that you want to include in your directory, you should perform a site survey of your data stores. That is, you should survey your enterprise for any and all relevant data. As part of this survey, you should:

- Locate all the organizations that manage your enterprise's information. Typically this will include your information services (IS), human resources (HR), payroll, and accounting departments.
- Identify the tools and processes that your enterprise uses to maintain this information. Some of the more common sources for information are networking operating systems (Windows NT, Novell Netware, Unix NIS), email systems, security systems, PBX [telephone switching] systems, and HR applications.
- Determine how centralizing each piece of data will impact the managing organizations. In the optimum case there is no impact, but you are likely to find that centralized data management might require new tools and new processes. Sometimes this centralization may require adding personnel to some organizations while reducing head count in others (in fact, overall you could see a reduction in head count as your processes become more efficient).

Because of the number of organizations that can be affected by the directory, it may be helpful to create a directory deployment team that includes representatives from each affected organization. For example, a corporation is likely to have a human relations (HR) department, an accounting and/or accounts receivable department, one or more manufacturing organizations, one or more sales organizations, and one or more development organizations. Including representatives from each of these organizations can help you to more rapidly perform the survey. More importantly, it always helps to listen to your users. Directly involving all the affected organizations can go a long way to building acceptance for the migration from local data stores to a centralized directory service.

Finally, you may need to run more than one site survey. This is especially true of large enterprises with offices in multiple cities or even countries. You may find your informational needs to be so complex that you will have to allow various organizations to master information at a local office rather than at a single, centralized master server. In this case, each office responsible for mastering information should run its own site survey. After this process has

been completed, the results of each survey should be returned to a central team (probably consisting of representatives from each office) for use in the design of the enterprise-wide data schema model and directory tree.

Schema design is discussed in Chapter 4, “Planning Directory Schema.” Directory tree design is discussed in Chapter 6, “Directory Tree Design.”

Analyzing Your Site Survey

Once you have located all the data important to your enterprise, you must determine if the data really can or should be stored in your directory. You must also determine whether all the people and applications that require access to the data are capable of reading from and/or writing to the directory. As an example, you may want to store every employee’s home address in the directory, but if your financial applications are unable to retrieve this information, then you may have to manage the information in multiple locations.

The decision about what types of data are maintained in your directory, and when you will start maintaining it there, will be driven by several factors:

- The data required by your various legacy applications (such as existing email applications) as well as your user population.
- The ability of your legacy applications to communicate with an LDAP directory service.

Your data analysis will involve determining the location where data will be mastered, who will own that data, and who can read that data. You should be careful to document your decisions. These activities are described in the following sections.

Data Mastering

Data mastering is important only if your data resides in more than one physical location. This can happen if you are using replication or if you are using applications that cannot communicate over LDAP.

Data Mastering for Replication

If you use replication, then you must consider which server will master, or supply, any given piece of data. This is because the Netscape Directory Server requires you to master any given piece of information on a single master server. (For more information on replication, see Chapter 7, “Planning Replication.”)

In the simplest case, you can choose to master all your data on a single Directory Server and then replicate that data to one or more consumer servers. In more complex cases, especially for large, multinational enterprises, you may want to master the data in a location close to the people, applications, or things that the data represents.

Data Mastering Across Multiple Applications

Data mastering will be an issue for you if you have applications that cannot directly communicate with the directory service. In this case, it best to keep the processes by which data can be changed, and the locations that it can be changed from, as simple as possible. Also, once you settle on a single location to master a piece of data, such as an employee’s name, then if possible use that same location to master all of the other data also contained there, such as the employee’s home address and telephone number. This allows you to more easily know where the ultimate repository for the information lies in the event that your databases get out of synch across your enterprise.

The approach that you take for data mastering can be one of the following:

- Master the data in both the directory service and all the other applications that are not directory-capable. This is the simplest case to implement because it does not require that you write custom scripts to move data in and out of the directory and the other applications. Of course, this method also means that if the data changes in one location, someone has to change it in all the other locations. Ultimately this is not an ideal situation because it will result in data being out of synch across your enterprise (which is what your directory is supposed to prevent).

- Master the data in the directory service and then write scripts or code to export changes to the relevant applications. This strategy makes the most sense if you have only a few applications that cannot communicate with the directory.

- Master the data in some application other than the directory and then write scripts, programs, or gateways to import that data into the directory. This makes the most sense if you can identify one or two applications that you already use to master your data, and you want to use your directory service only for lookups, such as is the case with online corporate telephone books.

The strategy that you choose depends on your enterprise's specific requirements. However, regardless of the approach that you choose, you are strongly recommended to keep your approach simple and to be consistent. You should not, for example, attempt to master data in multiple locations, then automatically exchange data between competing applications. Doing so will lead to a "last change wins" scenario and may increase your administrative overhead.

For example, suppose you want to manage an employee's home telephone number. This information is stored in both the LDAP directory as well as in an human resources (HR) database. Depending on the HR application, you can probably write an automatic application that transfers data from the LDAP directory to the HR database, and vice versa. However, if you attempt to master changes to that employee's telephone number in both the LDAP directory and the HR data, this can lead to a situation in which the last location that the telephone number was changed overwrites the information in the other database. This is acceptable as long as the last writing application had the correct information. But if that information was old or out of date (perhaps because, for example, the HR data was reloaded from a backup), then the correct telephone number in the LDAP directory will be destroyed.

For a brief look at writing custom filters, see Chapter 10, "Extending Your Directory Service."

Data Ownership

Data ownership refers to the person or organization that is responsible for making sure the data is up-to-date. As you plan your data management, you need to decide who you want to be able to write to the data. Some common strategies are:

- Allow read-only access to the directory for everyone except a small group of directory content managers.

- Allow individual users to manage some strategic subset of information for themselves. This information might include their own passwords, descriptive information about themselves and their role within the organization, their automobile license plate number, and contact information such as telephone numbers or office numbers.
- Allow a person's manager to write to some strategic subset of that person's information, such as contact information or job title.
- Allow an organization's administrator to create and manage entries for that organization. This effectively causes your enterprise's administrators to also be your directory content managers.
- Create groups that define roles, such as Human Resources, Finance, or Accounting. Allow each of these roles to have read and/or write access only to the data, especially sensitive data, that is needed by the group, such as salary information, government identification number (in the US, social security number), and home phone numbers and address.

As you perform this analysis and determine who can write to the data, you may find that multiple individuals need to have write access to the same information. For example, you will want some information systems (IS) or directory management group to have write access to employee passwords. You may also want the employee themselves to have write access to their own passwords. While it is generally necessary to allow multiple people to have write access to the same information, you should try to keep this group small and easily identifiable. Doing so will more easily allow you to ensure your data's integrity.

For information on setting access-control for your directory, see Chapter 5, "Planning Security Policies."

Determining Data Access

Besides determining data ownership, you must also decide who can read each piece of information. For example, you may decide to store an employee's home phone number in your directory. This can be useful information for a number of organizations, including the employee's manager and your human resources organization. Presumably, you would also want the employee herself to be able to read this information for verification purposes. However, home

contact information can be considered sensitive information. Therefore you must ask yourself (and your users) if you want this kind of data to be widely available across your enterprise.

Consequently, for each piece of information that you store in your directory, you must decide the following:

- Can the data be read anonymously? Anonymous access is supported by the LDAP protocol, and it is frequently used to allow easy lookups for commonly needed information such as office locations, email addresses, business telephone numbers (voice and fax), and so forth. The downside to anonymous access is that can access the directory can also access the information. Consequently, you should use this feature sparingly.

- Can the data be read widely across your enterprise? You can set up access-control so that it is necessary to log in to (bind to) the directory in order to read specific information. You might want to choose this form of general access over anonymous access to ensure that only members of your enterprise can view directory information. This form of access-control also allows you to see who is accessing any given piece of information because the user's login information can be captured in the directory's access log.

- Is there an identifiable group of people or applications who need to be able to read the data? Anyone who has write privileges to the data generally also needs read access (the exception to this is write access to passwords). However, you may also have data that is needed only by a specific organization or project group. Identifying these access needs as early as possible will help you to determine what groups and what access-controls you need to create in your directory.

  For information on group management, see "Planning Your Groups" on page 71.

As you make these decisions for each piece of directory data, you are essentially defining a security policy for your directory. The decisions that you make here will be heavily affected by the nature of your site and the kinds of security already available at your site. For example, if your site has a firewall or no direct access to the Internet at all, you may feel freer to support anonymous access than if you are placing your directory directly on the Internet.

The creation of a security policy and the way in which you implement it is described in detail in Chapter 5, "Planning Security Policies."

Documenting Your Site Survey

Because of the complexity of this planning activity, it is important that you document the results of your data analysis. One way to approach this problem is to create a simple table that outlines your decisions and outstanding concerns. You can build this table with the word-processing package of your choice, or you may want to use a spreadsheet so that the table's contents can easily be sorted and searched.

The following table is a simple example of what you might want to build to help you with your data planning. The table identifies data ownership and data access for each piece of identified data.

Data Name	Owner	Master Server/Application	Self Read/Write	Global Read	HR Writable	IS Writable
Employee name	HR	People Soft	Read-only	Yes (anonymous)	Yes	Yes
User password	IS	Directory US-1	Read/Write	No	No	Yes
Home phone number	HR	People Soft	Read/Write	No	Yes	No
Employee location	IS	Directory US-1	Read-only	Yes (must login)	No	Yes
Office phone number	Facilities	Phone switch	Read-only	Yes (anonymous)	No	No

For example, the directory must identify an employee's name. Each column in the table represents the following about employee names stored in the directory:

- Owner—The owner of this information is human resources (they are the organization responsible for updating or changing the information).
- Master Server/Application—The application that will manage employee name information is a HR application called People Soft.
- Self Read/Write—A person can read their own name, but not write (or change) it.

- Global Read—Employee names can be read anonymously by everyone who has access to the directory.
- HR Writable—Members of the group HR can change, add, and delete employee names in the directory.
- IS Writable—Members of the group called IS (information services) can change, add, and delete employee names in the directory.

Directory Schema Design

A final aspect to your data analysis is designing your directory schema. Essentially, schema design involves mapping the pieces of data that you have discovered to an appropriate attribute name and syntax. You will also have to decide what types of entries you will contain in your directory (people, devices, organizations, and so forth) and this will, in turn, determine the actual attributes that you have available to you on any given type of entry.

Most likely you will have to extend the standard directory schema to support your enterprise's needs. Consequently, you should leave room in your data analysis table for identifying the attribute name and object class structure on which the specific piece of data will be represented. In addition, you may want to record other schema-related information such as the syntax used for a type of data, the object class used by the entry that the data will be stored on, and so forth.

The next chapter discusses directory schema, and the concepts of attributes, object class structures, and schema extension. In addition, "Customizing the Schema" on page 47 provides general advice for managing your directory schema.

Chapter

4

Planning Directory Schema

Schema is a term used to described the type, or kind, of data that you can include in a directory. When you perform your site survey, you generate a lot of information on the data that you want to keep in your directory. Once you have an understanding of what you want to put in your directory, you must decide how to represent it there.

In this chapter you will learn how data is stored in the directory, and how you can customize your directory service to contain information that is unique to your enterprise. This chapter contains the following sections:

"Directory Data Representation Overview" on page 42—This section describes how data is stored in the directory through the use of attribute-data pairs.

"Object Classes" on page 43—This section describes the concept of and uses for object classes.

"Attributes" on page 44—This section describes the concept of and uses for object classes.

"Schema Checking" on page 47—This section describes the concept of schema checking.

"Customizing the Schema" on page 47—This section describes how and why you would extend your directory's schema.

Directory Data Representation Overview

Directory data is represented as attribute-data pairs. That is, any specific piece of information is associated with a descriptive attribute. For example, a person's name in the directory is stored using the `commonName`, or `cn`, attribute. Therefore, an entry that represents the person named Babs Jensen has the attribute-data pair of:

```
cn: Babs Jensen
```

In fact, the entire entry is represented as a series of attribute-data pairs. Babs Jensen's entry might, for example, look something like the following:

```
dn: uid=bjensen, ou=people, o=airius.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Babs Jensen
sn: Jensen
givenName: Babs
givenName: Barbara
mail: bjensen@airius.com
```

Notice in the previous example that multiple instances of a specific attribute are allowed. That is, `givenName` is used twice, each time with a unique value. This allows you to specify multiple values for a specific attribute.

The use of attribute-data pairs means that every entry in the directory is made up of multiple attributes. Further, you tell the directory what attribute values are allowed or required on the entry by defining one or more object classes for the entry (object classes are discussed in detail in the following section). By placing an object class on an entry, you are telling the Directory Server that the entry *can* have a certain set of attribute values, and *must* have another, usually smaller, set of attribute values.

Finally, before you can use an object class or an attribute in your directory, it must be identified to your directory. The total set of object classes and attributes known to your directory is referred to as your directory's *schema*. Your Directory Server comes with a standard schema that includes hundreds of object classes and attributes. In addition, you can extend this standard schema to represent information unique to your enterprise.

The following sections define these concepts in greater detail.

Object Classes

Object classes define the types of attributes an entry can contain. Most object classes define a set of required and optional attributes. This attribute list represents both required and allowed data that you can store on the entry. (For more information on attributes, see “Attributes” on page 44.)

For example, if you define an entry to use the `organizationalPerson` object class, then the `commonName` (`cn`) and `surname` (`sn`) attributes are required for the entry (you must specify values for these attributes when you create the entry). In addition, there is a fairly long list of attributes that you can optionally use on the entry. This list includes such descriptive attributes as `telephoneNumber`, `uid`, `streetAddress`, and `userPassword`.

The use of object classes and the concept of defining the type of data that you can store on your directory entry are meaningful only if schema checking is turned on. For information on schema checking, see “Schema Checking” on page 47.

Standard Object Classes

As is the case for all of the Netscape Directory Server’s schema, object classes are defined and stored directly in your Directory Server. This means that you can both query and change your directory’s schema by using standard LDAP operations.

Your directory recognizes a standard list of object classes by default. These are described in the *Netscape Schema Reference Guide*. You can find it at the following location:

`http://home.netscape.com/eng/server/directory/schema/`

Inheritance

Object classes are meant to use inheritance to define the total list of attributes that are either required or allowed on the entry. This inheritance is defined in the form of an object class structure. The structure begins with `objectClass top` and proceeds through a series of object class definitions, each of which adds to the list of required or allowed attributes. An object class that is meant to

be at a lower end of the structure should not be placed on an entry until all of that object class's antecedent object classes have also been defined on the entry.

Note While the LDAP specs call for an object class structure, the Netscape Directory Server does not currently enforce it. However, there is no guarantee that the Directory Server will not enforce these structures in the future. Therefore, you should always conform to the object class structure when designing and populating your directory.

For example, a person entry is usually defined with the following object class structure:

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

`objectClass top` is what allows additional object classes to be placed on the entry. `objectClass person` defines two required attributes (`commonName` and `surname`), and then a few optional attributes. `organizationalPerson` and `inetOrgPerson`, in turn, add more attributes to the list of optional attributes. Further, before you can put `inetOrgPerson` on the entry, you must first put object classes `top`, `person`, and `organizationalPerson` on the entry.

Attributes

Attributes hold information about a specific descriptive aspect of the entry. Each attribute consists of an attribute type and one or more attribute values. The attribute type identifies the class of information given by that attribute (for example, telephone number). The attribute value is the particular instance of information appearing in that entry (for example, 555-1999).

Attributes generally have short, mnemonic names. For example, `cn` is the abbreviation for `commonName`.

Required Versus Optional Attributes

Object class definitions usually contain at least one required attribute, and they always contain one or more optional attributes. If an object class requires an attribute, then you cannot add an entry using that object class to the directory unless the required attribute is also defined for the entry. Doing so results in an object class violation, and the add operation for the entry fails.

Optional attributes are attributes that you are allowed, but not required, to add to the directory entry. If you attempt to add an attribute to an entry that is neither required nor allowed according to the entry's object class definition, then the Directory Server returns an object class violation.

The concept of a required or allowed attribute is meaningful only if schema checking is turned on. If schema checking is turned off, then you can add any attribute to your directory; the Directory Server will make no attempt to ensure that the data you are adding to your directory conforms to the schema.

It is strongly recommended that you run your Directory Server with schema checking turned on. For information on schema checking, see “Schema Checking” on page 47.

A Note About Attribute Values

Perhaps one of the more confusing aspects about an LDAP schema is that you can generally place any data that you want on any attribute value. LDAP and the Directory Server place no restrictions on the data format, length, or type that you associate with individual attributes (other than some minimal syntax definitions that are used strictly for searching and pattern-matching purposes). For example, when you represent a telephone number in your directory tree, you can legally represent it in any of the following ways:

```
(408) 555-5555

408-555-5555

1-408-555-5555

1.408.555.5555

555-5555
```

This data model provides you with maximum flexibility when you design your data policy. However, remember that while the Directory Server and the LDAP protocol do not care about your data format, your LDAP clients probably do care about the representation of the data in your directory tree. This is especially true of graphical user interface clients that must display directory data in rigid-length fields.

The format of your directory data is also important to the user who is searching your directory tree for a particular value. For example, if the user is searching for an address such as

```
1206 Directory Drive
```

but you have stored the address in the form

```
1206 Directory Dr.
```

then the user may have difficulty locating the actual entry for which she is searching. For these two reasons, it is important that you be as consistent as possible when storing data in your directory tree. Pick a format and always follow it. When managing the data in your directory tree, remember what your LDAP clients and directory users expect to find there. Doing so will vastly improve the usability of your directory service.

In addition, the LDAP protocol may have defined certain attributes to contain data of a particular format. While the Netscape Directory Server does require you to represent this data in the proper format, your LDAP clients will likely be confused if they do not receive data in this defined format. One example of this is the `postalAddress` attribute, which expects an attribute value in the form of a multiline string that uses dollar signs ($) as line delimiters. For example:

```
postalAddress: 1206 Directory Drive$Pleasant View, MN$34209
```

Schema Checking

Schema checking causes the Directory Server to ensure that the attributes required for an object class are contained in the entry and that only attributes allowed by the object class are contained in the entry.

You can turn schema checking off, but this mode of operation is not recommended. Schema checking helps ensure that you are using a consistent schema, which is important for allowing an LDAP client to successfully search your directory. At a minimum, schema checking ensures that you have not misspelled any object class or attribute names.

For information on turning schema checking on and off, see the *Netscape Directory Server Administrator's Guide.*

Customizing the Schema

Regardless of the type of information that you contain in your Directory Server, you should never change the standard schema. If this standard schema proves to be too limiting for your uses, you should extend it to support your unique requirements. You should never delete standard schema elements from your server, nor should you ever completely replace the standard schema with a schema of your own design. Doing so can lead to interoperability problems with other directory services or other LDAP clients.

To help you extend your schema, the Netscape Directory Server includes a schema management tool. For more information, see the *Netscape Directory Server Administrator's Guide.*

When You Should Extend Your Schema

While the object classes and attributes supplied with the Directory Server should meet most of your needs, you will probably find that a given object class does not allow you to store specialized information about your organization. Or you may be using the Directory Server to support an LDAP-capable application, in which case you will need to extend your schema to support the object classes and attributes required by the application's information needs.

When to Add New Object Classes and Attributes

Add new attributes only when you add new object classes. If you find that you need to add an attribute to a standard LDAP object class, then you should:

1. Create a new object class. Define its parent object class to be the object class on which you wanted to add the attribute.
2. Add the new attribute on the new object class.

You should do this whenever you find that an existing object class structure does not support every kind of information that you want to store in your directory. Most frequently this occurs when you find that you want to store more information on a person entry than the `person`, `organizationalPerson`, or `inetOrgPerson` object classes support.

For example, you might want to store birth dates in your directory. No attribute for this information exists within the standard Netscape Directory Server schema, so you may choose to create a new attribute called `dateOfBirth` and allow this attribute to be used on entries representing people.

You should always look for an existing attribute that meets your needs before you extend your schema to include new attributes and object classes.

Object Class Strategies

There are two approaches you can take to new object class creation:

- You can create many new object classes, one for each object class structure that you want to add attributes to.
- You can create a single object class and use it to allow all of the attributes that you create for your directory. This object class can be used on any object class structure. You create this kind of an object class by defining its parent to be `objectClass top`.

For example, suppose your site wants to create the attributes `dateOfBirth`, `preferredOS`, `buildingFloor`, and `vicePresident`. Then you can create several object classes to allow the usage of some subset of these

attributes. For example, you might create an object class called `airiusPerson` and have it allow `dateOfBirth` and `preferredOS`. The parent of `airiusPerson` could be `inetOrgPerson`.

Note You should avoid requiring attributes on new object classes if possible. Requiring attributes tend to make your schema inflexible. It is best to allow rather than require attributes when you create new object classes.

You might also create an object class called `airiusOrganization` and have it allow `buildingFloor` and `vicePresident`. The parent of `airiusOrganization` might be the organization object class.

For example:

```
airiusPerson
  superior
    inetOrgPerson
  allowed
    dateOfBirth,
    preferredOS

airiusOrganization
  superior
    organization
  allowed
    buildingFloor,
    vicePresident
```

Alternatively you can create a single object class that allows all of these attributes and use it with any entry on which you want to use one or more of these attributes. It is best not to define any required attributes for this kind of an object class because you cannot be sure that you will want to always use a given attribute for every entry on which the object class will be included. For example:

```
airiusEntry
  allowed
    dateOfBirth,
    preferredOS,
    buildingFloor,
    vicePresident
```

Both strategies work equally well, and neither seems to offer a clear administrative advantage over the other. However, depending on your situation and your personal preferences, you will probably find one approach works a little better for you. Some things to consider are:

- Multiple object classes will result in more schema elements that you must create and maintain. However, the total number is likely to be reasonably small in even the most complex situations (one for each kind of object class structure that you might be using), and once the object classes are created there is very little administrative overhead to managing them. Still, you may find that it is easier for you to use a single object class if you find yourself considering adding more than two or three object classes to your schema.
- The multiple object class approach requires that you to be more careful and a lot more rigid in your approach to data design. This is because a structured approach forces you to consider the object class structure on which every piece of data will be placed. Depending on your personal preferences, you will find this to be either helpful or cumbersome. Pick the approach that works best for you.
- Using a single object class for everything is very useful if you find that you have data that you want to put on more than one type of object class structure. For example, suppose you want `preferredOS` on both a person and a group entry. Then you may want to create only a single object class to allow this attribute.
- There are no prohibitions, either technically or administratively, to mixing the strategies. If you find, for example, that it is best to create a few structural object classes, as well as another single object class to be used everywhere, then you should do that. There is nothing in the protocol or in the Netscape implementation that prevents you from doing this.

A Consistent Schema

It is important that you use a consistent schema within your Directory Server because LDAP clients locate entries in your directory by searching for object classes or attributes and their associated values. If you use an inconsistent schema, then it becomes very difficult to locate information in your directory tree efficiently.

An example of an inconsistent schema is a situation where you use an attribute to store a specific kind of information, and then later use a different attribute to store the exact same kind of data. Another example of an inconsistent schema is if you use multiple formats to store the same kind of information. That is, do this:

```
telephoneNumber: +1 555 123-4567
telephoneNumber: +1 555 123-2468
telephoneNumber: +1 555 123-3579
telephoneNumber: +1 555 123-1470
```

Do not do this:

```
telephoneNumber: (555) 123-4567
phone: +1 555 123-2468
phoneNumber: 123-3579
telephone: 1 555 123 1470
```

Further, LDAP clients are designed to work with a specific, well-defined schema. For the most part, the schema that these LDAP clients are designed to work with is the standard LDAP schema which is based on the X.500 standard. For this reason, most LDAP-based directory services begin with the standard LDAP schema, but then the schema is extended as the site discovers site-specific needs that are not met by the standard schema.

The Netscape LDAP schema takes this approach as well. Most of the schema elements are based on the LDAP standard. However, Netscape has extended the schema to allow for directory activities that are commonly performed by Netscape's customers. In addition, Netscape ships LDAP clients that expect the schema to appear as defined by Netscape.

You can enforce your Directory Server's schema by turning schema checking on. When schema checking is turned on, you cannot add an object class or attribute to your Directory Server that does not conform to the schema. For more information on schema checking, see the *Netscape Directory Server Administrator's Guide* (`http://home.netscape.com/eng/server/directory/4.1/admin/`).

Schema Extension Frequently Asked Questions

These are common questions that people new to the Directory Server tend to ask when they first extend their schema:

I want to represent *x* in my directory. Is there a standard attribute that I can use to do this?

What you should do when you are looking for an attribute for use with your directory is to first become familiar with the more common attributes as defined in the *Netscape Schema Reference Guide*.

If none of those common attributes seem to be appropriate for your needs, then try browsing through the total list of attributes using the Directory Server schema management tool. To narrow your search, consider the type of entry that the data represents (a person, an organization, a device, and so forth), and then examine those object class structures that define that type of an entry.

If you still cannot find a likely candidate, then extend your directory schema to support your informational needs. After all, if a standard attribute does actually exist that could support you, then it is probably so obscure or poorly named that few people will actually expect to find it in your directory.

Should I delete unused attributes/object classes from my schema?

In short, no. Unused schema elements represent no operational or administrative overhead for your directory. You gain nothing by deleting them, and you potentially create future coexistence problems by supporting only a subset of the standard LDAP schema.

Of course, if you have defined a new schema element for your directory and you have found that you are not using it, then by all means delete it from your schema. (However, first make sure that your directory data is not using it!) No harm can come from removing unused schema elements from your directory that are specific to your enterprise.

I am building a global directory, and I have data that I use in one part of the world that is not used in another part of the world. Do I have to make sure my schema extensions are defined worldwide?

It is not strictly necessary to use the same schema enterprise-wide, but in practice it is a very good idea to do so. This is because to support a large-scale directory, you are using replication to move data between servers. This means that whenever some piece of data is replicated to a server, that server must recognize the attribute used to represent the data. If the consuming server does not recognize the attribute, then the replication operation fails and the servers in your directory become out of synch.

The most likely reason why you might be replicating data to a server that does not recognize the schema used by that data is if you defined new data in your supplier server and you forgot to update the schema in your consumer server.

Remember that your goal should be to make your directory easy to administer. It is best to ensure that your schema is defined consistently across your entire enterprise, just in case you one day want to replicate a piece of data that you had not originally intended to replicate.

Chapter

5

Planning Security Policies

Part of any directory deployment is determining the security policy that you will use for your directory. *Security policy* means the rules that you use to grant and restrict directory access to individual users or, more frequently, groups of users.

In this chapter you will learn about the implementation of security policies and how you can represent security policies in your directory. This chapter includes the following sections:

"Security Policy Overview" on page 56—This section introduces the concept of security policies.

"Directory Access Rules" on page 57—This section describes how users access (or bind to) the directory. Topics include general client authentication, certificate-based authentication, authentication when using Netscape Directory Server for NT, anonymous access, and root DN authentication.

"Access Control Lists" on page 61—This section briefly introduces the concept of access control lists (ACL).

"Setting Permissions" on page 62—This section provides advice on when and where in your directory you should place access control information.

"The ACI Format" on page 66—This section provides a high-level overview to the Netscape Directory Server access control mechanism. Topics include ACI format and the types of permissions that you can define in your directory using ACIs.

"Creating a Security Policy" on page 71—This section describes the most important issues in building a security policy. Topics include using groups to manage access control, determining your directory's general access mechanism, and physical points of access to your directory.

Security Policy Overview

The security policy that you use for your directory is a reflection of the:

- Kind of information you are keeping in your directory
- Ways in which you are accessing your directory
- Ways in which you are updating or managing your directory

You can keep your directory as secure or relaxed as is dictated by the situation in your enterprise. The planning that you performed for your directory data will form the foundation for your security policy. In addition, the general attitude at your enterprise regarding security will also drive the nature of your directory's security policy.

The Netscape Directory Server provides an extremely flexible access control mechanism that you can use to define virtually any security policy, from the very simple to the very complex.

Because of the flexibility of the access control mechanism, plan some time during your deployment phase during which you can apply your access control information and make sure that it adequately represents your security desires. During this time, become familiar with the intricacies of the access control mechanism and make sure that the access controls that you add to your directory are as easy to administer as possible.

Your security policy should be strong enough to prevent sensitive information from being modified or retrieved by unauthorized users while simple enough that you can easily administer it. Ease of administration is very important when it comes to your security policy. A complex security policy can lead to mistakes

that either prevent people from accessing information that you want them to access or, worse, allow people to modify or retrieve directory information that you do not want them to access.

Directory Access Rules

It is useful to understand how LDAP clients bind (authenticate) to the directory before examining how you can control access to your directory.

The mechanism used for directory authentication is the same for both people and LDAP-aware applications (such as any Netscape server that is configured to use the directory).

User Authentication

In general you must authenticate to the Directory Server before you can access the directory's contents. The only exception to this is if you have set up anonymous access for your directory.

Conceptually, directory authentication can be thought of as logging in to the directory. LDAP Directory Servers, however, usually refer to this operation as *binding to* the directory.

Generally, bind operations consist of providing the equivalent of a user ID and a password. However, in the case of an LDAP directory, the user ID is a distinguished name. The distinguished name used to access the directory is referred to as the *bind DN*. It is generally true that the bind DN corresponds to the name of an entry in the directory, although this is not always the case. When you bind as the root DN, for example, there is no corresponding directory entry.

Also, the bind DN most frequently corresponds to an entry that represents a person, but again this is also not always true. Some directory administrators find it useful to bind as an organizational entry rather than as a person. The only real requirement is that the bind DN and the corresponding password must somehow be known to the directory. This means that the entry represented by the bind DN must use an object class structure that allows the `userPassword` attribute.

Most LDAP clients go to some lengths to hide the bind DN from the user because DNs are often long strings that are hard to remember and easy to mistype. When a client attempts to hide the bind DN from the user, it uses a bind algorithm such as the following:

1. The user enters a unique identifier such as a user ID (For example, `fchen`).
2. The LDAP client searches the directory for that identifier and returns the associated distinguished name (such as `uid=fchen, ou=people, o=airius.com`).
3. The LDAP client binds to the directory using the retrieved distinguished name and the password supplied by the user.

Both the Directory Server gateway and the administration server Users and Groups gateway use this type of authentication mechanism to bind the user to the directory. In addition, any Netscape server that is using the directory for user-authentication purposes also follows this algorithm.

Certificate-Based Authentication

An alternate form of directory authentication involves using security certificates to bind to the directory. Instead of providing a user name and a password, the user is prompted for a password the first time he attempts to access the directory. However, this password is not matched to a password stored in the directory. Instead, it is the password that opens the user's certificate database.

If the password supplied by the user is correct, the directory client obtains authentication information from the certificate database. The client and the Directory Server then use this information to identify the user by mapping the user's certificate to a directory DN. Directory access is allowed or denied based on the directory DN that is discovered based on this identification process.

For more information on certificates and SSL, see *Managing Servers with Netscape Console*.

Authenticating Using Directory Server for NT

If you are not using the NT Synchronization Service, you can skip this section.

The Directory Server is capable of performing NT pass-through authentication if all the proper conditions are met. *NT pass-through authentication* is the process by which the Directory Server contacts an NT domain and confirms a user's authentication with that domain. If the user's authentication credentials are confirmed by the NT domain, the user is granted access to the directory.

When users authenticate to a Directory Server running on NT, the Directory Server first attempts to confirm the user's identity using the normal Directory Server authentication mechanisms. If this authentication fails, the Directory Server attempts to confirm authentication with the appropriate NT Primary Domain Controller if the following conditions are true:

- The Directory Server is running in the NT domain where the authentication must occur, or the Directory Server is running in an NT domain that shares a trust relationship with the NT domain where the authentication must occur.
- The user's bind attempt must provide a distinguished name that is known to the directory. This is the bind DN, and it is used to retrieve the user's bind entry.
- The password that the user provides must not match the password stored on the user's bind entry. This condition can also be met if the user's bind entry does not have a `userPassword` attribute-data pair.
- The user's bind entry contains the `NTUser` object class.

In the event that the previous conditions are met, the Directory Server:

1. Contacts the NT domain identified on the user's `NTUserDomainId` attribute.
2. Attempts to authenticate to the NT domain using the user ID stored on the `NTUserDomainId` attribute and the password provided by the user on the original authentication attempt.

If the NT pass-through authentication succeeds, then the user is granted access to the Directory Server. Access is granted based on the permissions granted for the user's bind entry.

Anonymous Access

Anonymous access can be configured for the directory such that anyone can access the directory without providing bind credentials. That is, clients do not need to provide a bind DN or password to gain access. When a bind DN and password are not provided to the directory, the directory assumes a null bind DN and a null password, and the access that it allows or denies depends on the permissions that you have set up for anonymous access.

Anonymous access is also granted if the user authenticates using a bind entry that does not include a `userPassword` attribute-data pair and the user does not provide a password on the bind attempt.

However, the Directory Server fails the bind attempt if the user authenticates using a bind entry that does not include a `userPassword` attribute-data pair and the user provides any non-null string for the password.

Usually directory administrators only allow anonymous access for read, search, and compare privileges (not for write, add, delete, or selfwrite). Also, the access is usually limited to a subset of attributes that contain information generally useful across the enterprise, such as person names, telephone numbers, email addresses, and so forth. Anonymous access should never be allowed for more sensitive data such as government identification numbers (social security numbers in the US), home telephone numbers and addresses, salary information, and so forth.

The various privileges that you can set for the directory are described in "Permissions" on page 67. For general advice for allowing or denying access to specific attributes, see "Setting Permissions" on page 62.

Directory Manager Authentication

The Directory Manager is the distinguished name for the privileged directory user (it was formerly known as the root DN). After authentication, the Directory Manager has complete access to the directory regardless of the access controls in use on the directory.

It is not required that a Directory Manager be configured for the directory. In general, though, the Directory Manager is configured when the Directory Server is configured. As a general rule, you should use your Directory Manager to initially populate your database and set up access controls for normal directory entries. As soon as you have configured at least one entry that has full access to the directory, then you should stop using the Directory Manager for normal directory operations.

Warning If you configure a Directory Manager at server installation time, you must also provide a Directory Manager password. However, it is possible for this password to be deleted from `slapd.conf` by direct editing of the file. This can create operational problems with your server. Always be sure a Directory Manager password is configured in `slapd.conf` when a Directory Manager is configured for your Directory Server.

For more information on database usage and your Directory Server, see the *Netscape Directory SDK Programmer's Guide*.

Access Control Lists

Access Control Lists (ACLs) provide you with the ability to control access to your directory. You can either allow or deny access to your directory using ACLs. Your directory's ACL is composed of a series of one or more access control information (ACI) statements that either allow or deny permissions such as read, write, search, and compare to specified entries and their attributes.

Using the ACL, you can set permissions for:

- the entire directory
- a particular subtree of the directory
- specific entries in the directory

- a specific set of entry attributes
- any entry that matches a given LDAP search filter

In addition, you can set permissions for a specific user, all users belonging to a specific group, or for all users of the directory. Finally, you can also define access for a network locations such as an IP address or a DNS name.

Setting Permissions

By default all users are denied access rights of any kind. The exception to this is the Directory Manager. For this reason, you must set some ACIs for your directory if you want your users to be able to access your directory.

The following sections describe the access control mechanism provided by your Directory Server. For information on how to set ACIs in your directory, see the *Netscape Directory Server Administrator's Guide*.

The Precedence Rule

When a user attempts any kind of access to a directory entry, the Directory Server examines the access control set in the directory from the entry being accessed back to the top, or root, or the directory tree held by the server.

Because the ACL allows you to set permissions that denies as well as grant access, it is important to remember the precedence rule:

Precedence Rule If two permissions exist and are in conflict, the permission that denies access always takes precedence over the permission that grants access.

For example, if you deny write permission at the directory's root level, and you make that permission applicable to everyone accessing the directory, then no user can write to the directory regardless of any other permissions that you may allow for a user. To allow a specific user write permissions to the directory, you have to restrict the scope of the original deny-for-write so that it does not include the user. Then you have to create an additional allow-for-write permission for the user in question.

Because of this, and because by default users are denied access anyway, you should use deny permissions sparingly in order to avoid potential confusion.

Allowing or Denying Access

You can either explicitly allow access or you can explicitly deny access to your directory tree. Be careful of explicitly denying access to the directory. Because of the precedence rule, if the directory can find any rules explicitly forbidding access, it will do so regardless of any conflicting permissions that may grant access. It can be very confusing to set a rule that allows access, only to find that access is not permitted because another rule explicitly denies the access from some other point in the directory.

Because of the precedence rule and the administrative problems an explicit deny can create, and because access is denied by default anyway, it is best to limit the scope of your allow access rules to include only the smallest possible subset of users or clients. For example, you can set permissions that allow users to write to any attribute on their directory entry, but then deny all users except members of the Directory Administrators group the privilege of writing to the `uid` attribute. A better approach is to write two access rules that allow write access in the following ways:

- Create one rule that allows write privileges to every attribute except the `uid` attribute. This rule should apply to everyone.
- Create one rule that allows write privileges to the `uid` attribute. This rule should apply only to members of the Directory Administrators group.

By providing only allow privileges that carefully restrict the scope of the privilege, you avoid the need to set an explicit deny privilege and therefore avoid possible administrative problems.

When to Explicitly Deny Access

Because of the reasons stated above, there are very few situations when it is desirable to set an explicit deny. Even so, you may find an explicit deny to be useful in the following situations:

- You have a large directory tree with a complicated ACL spread across it. For security reasons, you find that you suddenly need to deny access to a particular user, group, or physical location. Rather than spend the time to carefully examine your existing ACL to understand how to appropriately restrict the allow permissions, you may want to temporarily set the explicit deny until you have time to do this analysis. If your ACL has become this

complicated, then in the long run the deny ACI will only add to your administrative burdens. As soon as possible, you should rework your ACL to avoid the explicit deny and simplify your overall access control scheme.

- You want to restrict access control based on a day of the week or an hour of the day. For example, you can deny all writing activities from Sunday at 11:00 p.m. (2300) to Monday at 1:00 a.m. (0100). From an administrative point of view, it may be easier to manage an ACI that explicitly restricts time-based access of this kind than to search through the directory for all the allow for write ACIs and restrict their scopes in this time frame.
- You want to restrict privileges when you are delegating directory administration authority to multiple people. That is, if you are allowing a person or a group of people to manage some part of the directory tree, but you want to make sure that this delegated administration group does not modify some aspect of that part of the tree, then use an explicitly deny to disallow that type of access. For example, if you want to make sure the Mail Administrators do not allow write access to the common name attribute, then set a ACI that explicitly denies write access to the common name attribute.

Where to Place Access Control Rules

Access Control rules can be placed on any entry in the directory. For the most common usages of the Directory Server, access control rules are placed on entries of type `country`, `organization`, `organizationalUnit`, `inetOrgPerson`, or `group`.

To simplify your ACL administration, try to group these rules together as much as possible. Since a rule generally applies to its target entry and to all that entry's children, it is best to place access control rules on root points in the directory or on directory branch points, rather than scatter them across individual leaf (such as person) entries.

Using Filtered Access Control Rules

One of the more powerful features of the Directory Server ACI model is the ability to use LDAP search filters to set access control. This feature allows you to set a type of access to any directory entry that matches a defined set of criteria.

For example, you could allow read access for any entry that contains an `organizationalUnit` attribute that is set to Marketing.

One of the best uses of filtered access control rules is that you can use this feature to predefine types of access. For example, suppose your directory contains home address and telephone number information. Some people will want to publish this information, while others will want to be "unlisted." You can handle this situation by doing the following:

- Create an attribute on every user's directory entry called `publishHomeContactInfo`.
- Set an access control rule that grants read access to the `homePhone` and `homePostalAddress` attributes only for entries whose `publishHomeContactInfo` attribute is set to TRUE. Use an LDAP search filter to express the target for this rule.
- Allow your directory users to change the value of their own `publishHomeContactInfo` attribute to either TRUE or FALSE. In this way, the directory user can decide whether this information is publicly available.

For more information on using LDAP search filters, and on using LDAP search filters with ACIs, see the *Netscape Directory Server Administrator's Guide.*

The ACI Format

When designing your security policy, it is helpful to have a broad understanding of how ACIs are represented in your directory. It is also helpful to understand what permissions you can set in your directory. This section is not intended to provide you with a detailed description of the Netscape ACL mechanism. For a complete description of the ACL format, see the *Netscape Directory Server Administrator's Guide*.

Directory ACIs take the following general form:

```
<target> <permission> <bind rule>
```

Each of these parts of an ACI are:

- `<target>`—Specifies the entry (usually a subtree) to which the ACI is targeted, and/or the attribute to which it is targeted. That is, the `<target>` identifies the directory element that the ACI applies to. An ACI can target only one entry, but it can target multiple attributes. In addition, the `<target>` can contain an LDAP search filter. This allows you to set permissions for widely scattered entries that contain common attribute values.
- `<permission>`—Identifies the actual permission being set by this ACI. The `<permission>` says that the ACI is allowing or denying a specific type of directory access, such as read or search, to the specified `<target>`.
- `<bind rule>`—Identifies the bind DN or network location to which the permission applies. The bind rule may also specify an LDAP filter, and if that filter is evaluated to be true for the binding client, then the ACI applies to the client.

Taken together, ACIs are expressible as follows:

> "For the directory object `<target>`, allow or deny `<permission>` if the `<bind rule>` is true."

`<permission>` and `<bind rule>` are set as a pair, and you can have multiple `<permission>-<bind rule>` pairs for every target. This allows you to efficiently set multiple access controls for any given target. For example:

```
<target>(<permission><bind rule>)(<permission><bind rule>)...
```

Targets

You must decide what entry is targeted by each and every ACI that you create in your directory. If you target a directory entry that represents a directory branch point, then that branch point as well as all of its child entries are included in the scope of the permission. If you do not explicitly specify a target entry for the ACI, then the ACI is targeted to the directory entry that contains the ACI statement. Also, the default set of attributes targeted by the ACI is any attribute available in the targeted entry's object class structure.

For every ACI, you can target only one entry or only those entries that match a single LDAP search filter.

In addition to targeting entries, you can also target attributes on the entry. This allows you to set a permission that applies to only a subset of attribute values. You can target sets of attributes by explicitly naming those attributes that are targeted, or by explicitly naming the attributes that are not targeted by the ACI. Use the latter case if you want to set a permission for all but a few attributes allowed by an object class structure.

Permissions

You allow or deny permissions. In general, you should avoid denying permissions for the reasons explained in "Allowing or Denying Access" on page 63.

You can allow or deny the following permissions:

- Read—Indicates whether directory data may be read.
- Write—Indicates whether directory data may be changed or created. This permission also allows directory data to be deleted, but not the entry itself. To delete an entire entry, the user must have delete permissions.
- Search—Indicates whether the directory data can be searched. This differs from the Read permission in that Read allows directory data to be viewed if it is returned as part of a search operation. For example, if you allow searching for common names and read for a person's room number, then the room number can be returned as part of the common name search, but the room number cannot, itself, be searched for. This would prevent people from searching your directory to see who it is that sits in room number 12.

- Compare—Indicates whether the data may be used in comparison operations. Compare implies the ability to search, but actual directory information is not returned as a result of the search. Instead, a simple Boolean value is returned that indicates whether the compared values match. This is used most commonly to match `userPassword` values during directory authentication.
- Selfwrite—Used only for group management. This permission allows someone to add or delete themselves to or from a group.
- Add—Indicates whether child entries can be created. This permission allows a user to create child entries beneath the targeted entry.
- Delete—Indicates whether an entry can be deleted. This permission allows a user to delete the targeted entry.

Bind Rules

While targets indicate what directory object the ACI applies to, bind rules indicate the bind situations that the ACI applies to. Normally this means that the bind rule indicates which bind DNs the permission applies to.

For example, you can set a permission that allows anyone binding as Babs Jensen to write to Babs Jensen's telephone number. The bind rule in this permission is that part that states "if you bind as Babs Jensen." The target is Babs Jensen's phone number, and the permission is write access.

Bind rules allow you to easily express that the ACI applies only to a user's own entry. You can use this to allow users to update their own entries without running the risk of a user updating another user's entry.

However, bind rules can be used to express more complex bind situations than just bind DNs.

Using bind rules, you can indicate that the ACI is applicable:

- Only if the bind operation is arriving from a specific IP address or DNS hostname. This is often used to force all directory updates to occur from a given machine or network domain.

- If the person binds anonymously. Setting a permission for anonymous bind also means that the permission applies to anyone who binds to the directory as well.
- For anyone who successfully binds to the directory. This allows general access while preventing anonymous access.
- Only if the client has bound as the immediate parent of the entry.
- Only if the entry that the person has bound as meets a specific LDAP search criteria.

The following keywords are provided to help you more easily express these kinds of access:

- Parent—If the bind DN represents the immediate parent entry, then the bind rule is true. This allows you to grant specific permissions that, for example, allow a directory branch point to manage its immediate child entries.
- Self—If the bind DN is the same as the entry for which access is requested, then the bind rule is true. This allows you to grant specific permission that, for example, allows an individual user to update her own entry.
- All—The bind rule is true for anyone who has successfully bound to the directory.
- Anyone—The bind rule is true for everyone. This keyword is what allows or denies anonymous access.

Using ACIs: Some Hints and Tricks

The following are some ideas that you should keep in mind when you implement your security policy. They can help to lower the administrative burden of managing your directory security model and improve your directory's performance characteristics.

Some of the following hints have already been described earlier in this chapter. They are included in this list for completeness.

- Minimize the number of ACIs in your directory. You should use no more than 100 ACIs in your directory. This is because it is difficult to manage a large number of ACI statements. Specifically, a large number of ACIs will make it hard for you to determine immediately which directory object can be accessed by what client.

- Identify the smallest set of attributes on any given ACI. This means that if you are allowing or restricting access to a subset of attributes on an object, determine whether the smallest list is the set of attributes that are allowed or the set of attributes that are denied. Then express your ACI so that you are managing the smallest list.

  For example, the people object class structure contains dozens of total attributes. If you want to allow a user to update just one or two of these attributes, then write your ACI so that it allows write access for just those few attributes. If, however, you want to allow a user to update all but one or two attributes, then create the ACI so that it allows write access for everything but a few named attributes.

- Use LDAP search filters cautiously. Because search filters do not directly name the object that you are managing access for, their use can result in unexpected surprises, especially as your directory becomes more complex. If you are using search filters, make sure you test your directory access thoroughly each time you change the filters so that you are certain what the results of the changes mean to your directory.

- Avoid explicit denies. Because of the precedence rule, it is administratively easier to simply restrict the scope of any given allow access to the smallest set of possible users or directory objects. If you do use explicit denies, be prepared for situations where users are unexpectedly unable to access directory objects.

- Do not duplicate ACIs in differing parts of your directory tree. Watch out for overlapping ACIs. For example, if you have an ACI at your directory root point that allows a group write access to the `commonName` and `givenName` attributes and another ACI that allows the same group write access for just the `commonName` attribute, then consider reworking your ACIs so that only one control grants the write access for the group. As your directory grows more complicated, it will become increasingly easy to accidentally overlap ACIs in this manner. By avoiding ACI overlap, you will make your security management easier while potentially reducing the total number of ACIs contained in your directory.

- Name your ACIs. While naming ACIs is optional, granting each ACI a short, meaningful name will help you to manage your security model, especially when examining your ACIs from the Directory Server Manager.
- Group your ACIs as closely together as possible within your directory. Try to limit ACI placement to your directory root point and to major directory branch points. This will help you manage your total list of ACIs, as well as help you keep the total number of ACIs in your directory to a minimum.
- Use the root DN sparingly, if at all. Instead, you should create a directory administrator entry that has full access to your directory contents. This allows you to easily change or modify the administrative entry in the event that it becomes compromised (whereas changes to the root DN require a server restart).

Creating a Security Policy

Now that you have a general idea of how access control is handled in the directory, you can begin generating your security policy.

When you performed your data analysis (see **“Analyzing Your Site Survey” on page 34**), you should have made some basic decisions about who can read and write the individual pieces of data in your directory. This information now forms the basis of your security policy.

Planning Your Groups

When you perform your data analysis, you make some basic decisions on what groups of people can read and write information. You should now formally identify all the groups that you need in your directory. As you formally identify and name your groups, make sure you document your decisions. Make sure you know the formal name of each group, the general permissions you want each group to have, and the people who you immediately will want to add to each group.

Note While they are implemented in essentially the same way, there are two types of groups that your directory will contain. One is used for identifying access control permissions for a defined set of people. Another is used for managing mailing lists. This latter group typically uses the `mailGroup` object class. Its usage is not discussed in this manual.

For example, many directories commonly have an Administrator's group (Netscape servers actually require this group for certain kinds of server administration. For more information, see *Managing Servers with Netscape Console*). Usually the Administrator's group is granted complete access to the directory. By adding users to this group, you are granting those users full read, write, search, and compare privileges to the entire directory. Typically you will want this group to contain just a small number of users (3-10, depending on the size of your enterprise).

In addition to the Administrator's group, you may also want to create groups of users that have some, but not complete, write access to the directory. One such group may be Department Administrators. Some enterprises use Department Administrators to manage a subset of employee information, such as fax numbers, department numbers, and physical location information. By distributing these administration burdens to a select, identifiable group of people, you are make your directory easier to manage and ensure that you have data consistency at least at the department level.

Some other groups that you might need include:

- an Accounting group for people with read and write access to financial information
- an HR group for people with read and write access to sensitive employee information
- a Sales Administration group for people with read and write access to customer contact information
- a Mail Administrators group for people who can set up and manage mail accounts
- a News Administrators for people who can create and manage news (discussion) groups

In addition, if you want to allow local management of directory data, then you need to consider the administrative groups needed at each individual site. This is especially true if you are allowing directory data to be managed at local sites around your enterprise. This situation is especially likely to occur if you are replicating data to geographically separate locations.

Group Usage Advice

As you plan your groups, make sure you do the following:

- Avoid high levels of nesting within groups. That is, do not create a situation where one group is a member of another group, which is in turn a member of a third group, and so on. This kind of nesting will severely impact your server performance. You should limit yourself to no more than 2 levels of nesting to avoid server degradation.
- Do not create circular groups. That is, do not create a situation where group 1 is a member of group 2 and group 2 is a member of group 1. This can cause the server to loop repeatedly through the group attempting to find out if a bind DN is a member of the group. Eventually the server will time out, but activities like this will degrade your server's performance.

Determining General Directory Access

A basic decision you need to make regarding your security policy is how will users generally access the directory. In short, will you allow anonymous access, or will you require every person who uses your directory to bind to the directory?

Consider these key points:

- Anonymous access provides the easiest form of access to your directory. However, anonymous access does not allow you to track who is performing what kinds of searches; only that someone is performing searches. Also, remember that anonymous access means anyone can make a connection to your directory can access the data. Therefore, if you attempt to block a specific user or group of users from seeing some kinds of directory data, but you have allowed anonymous access to that data, then those users can still access the data simply by binding to the directory anonymously.

- Some LDAP clients, such as Netscape Communicator 4.x, require anonymous access for directory lookups. The address book function in Communicator 4.x requires anonymous access, as does the public key search function in Communicator 4.x. If you are using an LDAP product that requires anonymous access, then you should carefully consider what pieces of directory data are required by that product and make sure to allow anonymous access only to those pieces of data.

Determining Points of Access

Many enterprises restrict directory updates to clients running at a given IP address or from a specific DNS hostname. This ensures that the directory cannot be updated unless the person has access to a specific machine or sets of machines.

You may want to restrict read and search access to your directory to only a specific IP address, subnet, or domain name. This allows you to ensure that only people physically inside your enterprise can access your directory. To support this, the Netscape Directory Server ACL model allows wildcard patterns on IP addresses and DNS hostnames.

You may feel that this level of access control is unnecessary for a wide variety of reasons. However, you should consider this level of directory access if any of the following apply to you:

- You are running an extranet and are allowing some amount of directory information to be read and/or written to your directory from the Internet. In this case, you may want to restrict access to your directory based on domain name or IP address.
- You have one and only one location from which you want to update directory information. This will generally be true if you are populating your directory by using some HR or Accounting package, or by using some kind of an LDAP gateway. In this case, restrict directory updates to only that machine or subnet where the mastering application is running.
- You are an Internet Service Provider and you want to make sure directory updates and/or reads are performed only from the appropriate site.

Note There is no way to prevent people from bypassing this type of security by faking (spoofing) the IP address or DNS hostname that they are running from. Therefore, avoid relying only on physical points of access to determine access control unless you are confident that your user community will not attempt to bypass your directory security policy.

Chapter

6

Directory Tree Design

LDAP directory services organize directory entries using a directory tree. The design that you should use for your directory tree is determined by the types of information that you want to store in your directory service, the physical nature of your enterprise, the applications that you will use with your directory service, and the types of replication that you need to support your enterprise's activities.

In this chapter you will learn about directory trees and you will find advice on how you should design your directory tree. This chapter includes the following sections:

"Directory Tree Overview" on page 78—This section describes the concept of a directory tree. Also included is advice on how to plan and define your directory tree's suffix.

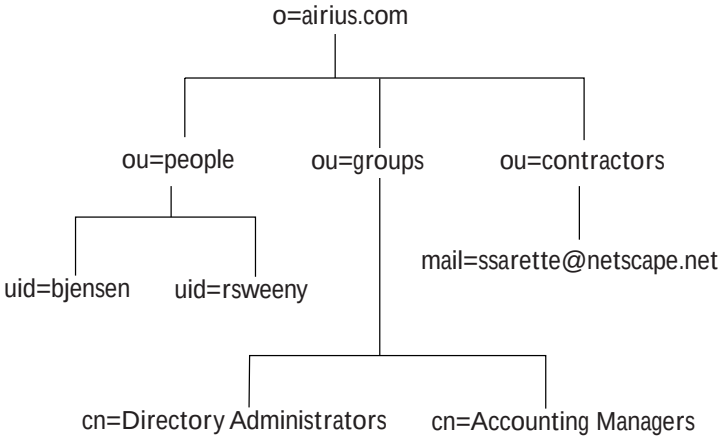
"Branching Your Directory Tree" on page 84—This section describes the branching strategies and the distinguished name designs that you should use with your directory tree.

"Naming Person Entries" on page 91—This section describes the distinguished name conventions that you should use for directory entries that represent people.

"Naming Non-Person Entries" on page 93—This section describes the distinguished name conventions that you should use for directory entries that do not represent people.

Directory Tree Overview

As is the case with many file systems, the directory tree is usually visualized as an inverted tree. The root of the tree is at the top and individual directory entries are at the lowest points on the tree. This manual refers to an entry that represents a new branch in the tree as a *branch point*. The following figure depicts a typical directory tree:

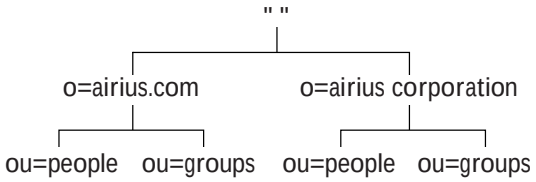


The root point of the tree is represented by a special entry whose distinguished name is called the *directory suffix*. An important part of your directory tree design will be selecting this DN. (This topic is discussed in the next section.)

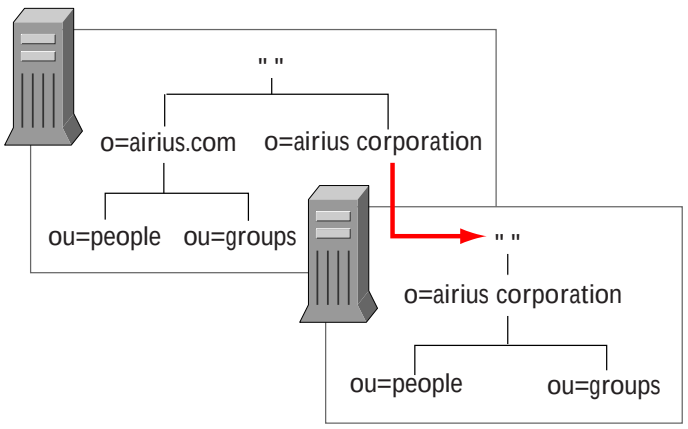
Your directory service can include multiple directory suffixes. Each directory suffix can be considered to represent a unique tree. Existing about these directory suffixes is a special entry called the *Root Directory Server Specific Entry* (DSE). The root DSE is a concept borrowed from X.500. The root DSE is a special entry that exists above the root entries in your directory service's various directory trees. It is represented by the null string (literally **""**), and it contains attributes specific to the Directory Server instance.

Multiple Suffixes

Each suffix that you use with your directory service represents a unique directory tree. There are two ways that you can include multiple trees in your directory service. The first is to physically include the individual trees in the database(s) served by your Directory Server.



The second is to use smart referrals to point directory clients to the server that physically contains the directory tree. A *referral* is a mechanism that allows a Directory Server to tell an LDAP client the location and name of the physical server where directory data can be found. Using smart referrals to support multiple directory trees requires that you configure your server with the suffixes used by the various directory trees, and also configure your Directory Server to contain the representative directory suffix entry:



Smart referrals are described in Chapter 8, “Planning Referrals.”

Planning Your Directory Suffix(es)

When you deploy a Netscape Directory Server, you will automatically have multiple suffixes defined for your directory. Most of these are used internally by the Directory Server and they require little or no planning on your part. These include the suffix for the machine data area, the suffix for the change log database (if configured), and the suffix for your directory schema. For more information on these standard directory trees, see the *Netscape Directory Server Administrator's Guide.*

Of particular importance to planning your directory tree is determining which suffix(es) you will use to store your directory data. In almost all circumstances, you should use only one suffix for data storage. However, in some specialized cases, such as for an extranet or for Internet Service Providers, multiple suffixes may be required to fully support your directory service.

In either case, the criteria for selecting a directory suffix is the same whether you will use one or 50 suffixes for your directory.

Selecting a Suffix

Traditionally LDAP directories have followed the X.500 naming convention for suffixes. This traditional form specifies an organization name followed by a country designation. For example:

```
o=Airius, c=US
```

This suffix pattern effectively roots your directory service in a specific country, which may or may not be appropriate depending on the nature of your enterprise.

The only time rooting your directory tree in a country designator is really necessary is when you are participating in an X.500 directory service. In that case, you should follow whatever DN conventions are in use by the X.500 service.

Generally, rooting your directory tree in a country designator is an acceptable practice if your enterprise is not multinational in nature. Therefore, if your enterprise is a government or military organization, educational institution, or small business, no harm can come from rooting your directory tree in a country designator. It is worth noting, however, that there is also no real benefit in

doing so. Unlike X.500 directory services, LDAP does not require any specific format for the DN naming conventions, so you are free to use whatever you want for your directory suffix.

The best practice for choosing a directory suffix is to align it with a DNS name or Internet domain name associated with your enterprise. This is the best approach regardless of the nature of your enterprise. Therefore, if your enterprise owns the domain name of `airius.com`, then you should use a directory suffix of:

```
o=airius.com
```

Another option is to use the `dc` (`domainComponent`) attribute to represent your suffix. To represent your domain name using the `dc` attribute, you break your domain name into its component parts as follows:

```
dc=airius, dc=com
```

Note If you are going to use a Netscape 1.x Certificate Server with your intranet, you should avoid the use of the `dc` attribute in your DNs because the 1.x certificate server does not support this attribute.

Using Multiple Suffixes: ISPs

For most directory services there is no need to use multiple suffixes with your Directory Server. The exception to this rule is Internet Service Providers (ISPs) who may be supporting multiple enterprises with their directory services. If you are an ISP, then you should approach each of your customers as a unique enterprise and design their directory trees following the advice and concerns as outlined in this book. For security reasons, each account should be provided a unique directory tree with a unique suffix and an independent security policy.

Your Directory Server can use multiple databases, so you can assign each customer their own database, and you may want place each directory tree on its own disk drive. Placing each directory tree in its own database allows you to perform backup and restore activities for each directory tree without impacting your other customer's operations.

By placing each directory tree on a unique disk drive, you reduce possible performance problems caused by disk contention, and you also reduce the number of accounts potentially impacted by a disk outage.

For more information on using backend plug-ins with your Directory Server, see the *Netscape Directory Server Programmer's Guide*.

Using Multiple Suffixes: Large Enterprises and Extranets

If you are supporting a very large enterprise or an extranet with your directory service, then you may want to split up the physical locations where directory data is stored and use smart referrals to locate the data. That is, place specific parts of your directory tree on individual Directory Servers, and then use smart referrals to tie the logical directory together.

There are several reasons why you may want to do this:

- Your enterprise contains many groups that own their own data.
- Your site is enabling an extranet.

Your enterprise contains many groups that own their own data.

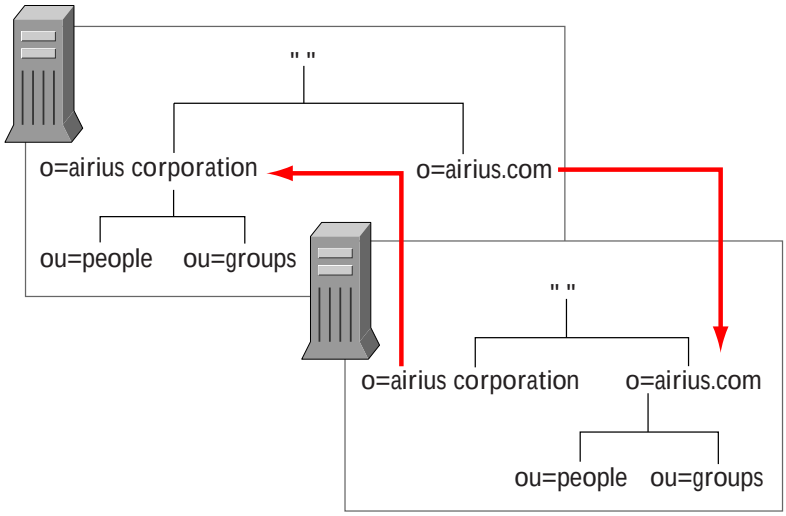
An example of this may be an enterprise that includes divisions or separate business units that are responsible for managing their own personnel information and/or email systems.

Your enterprise includes sites that are geographically distant from one another. Each site may include information that is mostly important to the people at that site, but may at times be of interest to other members of the enterprise.

One such example of this may be a consortium of international consulting firms. Each firm will maintain information that only it cares about, but it may also include information (such as telephone numbers, email addresses, mailing addresses, and other contact information) that may occasionally be of interest to members of other consulting firms. The nature of this information and the frequency by which it is queried by other members of the consortium may be such that replication of this information is not warranted.

In this situation, you can assign each consulting firm its own suffix. Once each organization has an assigned suffix (again, probably based on their Internet domain identity), you can configure each Directory Server in the global enterprise to recognize each of the participating suffixes. You should then create on each server a corresponding root directory entry for each suffix. Each such root directory entry should be a smart referral to the appropriate Directory Server.

For example:



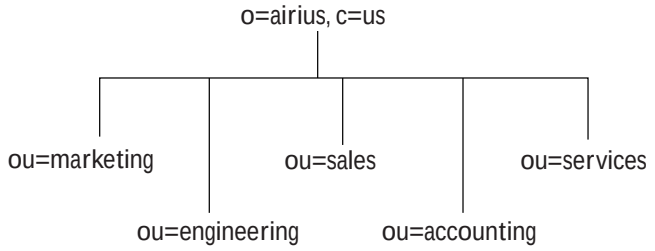
You could instead extend the directory tree so that it includes all members of the consortium, and then use smart referrals at each branch point to point to the appropriate Directory Server for the subtree. This method works best for tightly knit international enterprises where issues of organizational identity within the directory tree can be reduced or eliminated entirely. For enterprises that want to retain their suffix identification (this is usually purely a political and cultural issue), the use of smart referrals is the best approach

Your site is enabling an extranet.

In this case, you should configure your directory service to recognize the suffix used by each of your trading partners. Again, each suffix should have a corresponding root directory entry that is a smart referral to the directory service used by that particular trading partner. For more information on enabling extranets with your directory service, see “Enabling the Extranet” on page 139.

Branching Your Directory Tree

Traditionally, X.500 directory services branched their directory trees to reflect organizations across the enterprise. For example, if Airius Corporation had the organizational units of Marketing, Engineering, Sales, Accounting, and Services, then the traditional directory tree would be organized as follows:



Because X.500 preceded LDAP, there is strong momentum to continue designing directory trees along these traditional organizational lines. However, you should avoid this traditional directory tree structure. Most enterprises experience significant and frequent movement of personnel between organizations. Further, some enterprises, particularly corporations, often restructure and rename their organizations with some frequency. In the end, designing your directory trees along organizational lines will only cause you significant administrative burdens when these organizational lines change.

Branch Point Distinguished Name Attributes

As you decide how to branch your directory tree, you will need to decide what attributes you will use to identify the branch points. Remember that a DN is unique string composed of attribute-data pairs. Traditionally, DNs follow the format:

```
cn=<value>, l=<value>, ou=<value>, o=<value>, c=<value>
```

Netscape recommends that you change this traditional format to:

```
cn=<value>, l=<value>, ou=<value>, o=<domain>
```

The left-most attribute is called the Relative Distinguished Name (RDN). The appropriate value for the RDN is discussed in **“Naming Person Entries” on page 91** and **“Naming Non-Person Entries” on page 93**.

All of the intermediate attribute-data pairs represent branch points in your directory tree. This is an area where LDAP differs from X.500. While LDAP allows you to use any attributes and attribute values for your directory branch points, X.500 identifies which attributes can appear in a DN, and even the order in which they must appear. From the LDAP perspective there is not even a requirement to use any of the traditional attributes in your DN scheme.

However, there are some things you should consider when choosing attributes to represent your branch points:

- Be consistent. Some LDAP clients (and some directory administrators for that matter) may be confused if the DN format is inconsistent across your directory tree. That is, if `l` is subordinate to `o` in one part of your directory tree, then make sure `l` is subordinate to `o` in all other parts of your directory.

- Try to use only the traditional attributes (shown in Table 6.1 on page 86). While not strictly necessary, using the traditional attributes means that you have the greatest likelihood of retaining interoperability with third-party LDAP clients. Using the traditional attributes also means that they will be known to your directory schema, which is desirable when it comes time to build the entries associated with the DN.

- When creating entries, define the RDN within the entry. The RDN is the left-most DN attribute value. By defining at least the RDN within the entry, you can more easily locate the entry. Remember, you do not search against the actual DN; instead you search against the attribute values stored in the entry itself. For example, if you create a branch point called:

  ```
  ou=widget research, o=airius.com
  ```

  then add an `ou` attribute to the entry whose value is *widget research*.

- Attribute names have a meaning, so try to use the attribute name with the type of entry that they are representing. For example, do not use `l` (`locality`) to represent an organization, or `c` (`country`) to represent an organizational unit. The traditional attributes and their meanings are defined as follows:

Table 6.1 Traditional DN Branch Point Attributes

Attribute Name	Definition
c	Represents a country name.
o	Represents an organization name. This attribute is typically used to represent a large divisional branching such as a corporate division, academic discipline (the humanities, the sciences), subsidiary, or other major branching within the enterprise. You should also use this attribute to represent a domain name as discussed in "Selecting a Suffix" on page 80.
ou	Represents an organizational unit. This attribute is typically used to represent a smaller divisional branching of your enterprise than an organization. Organizational units are generally subordinate to the preceding organization.
st	Represents a state or province name.
l	Represents a locality, such as a city, country, office, or facility name.
dc	Represents a domain component as discussed in "Selecting a Suffix" on page 80.

Note A common mistake is to assume that you search your directory based on the attributes used in the distinguished name. This is not true. The distinguished name is only a unique identifier for the directory entry, and you cannot search against it. Instead, you search for entries based on the attribute-data pairs stored on the entry itself. Thus, if your distinguished name contains the value *cn=babs jensen*, then you can cannot search against that attribute-data pair unless you have defined `cn: babs jensen` in the directory entry.

Branching Strategies

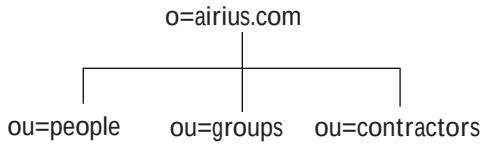
There are no performance or administrative benefits to a heavily branched directory tree. Further, when you search an LDAP directory service, you do so based on the attribute-data pairs used in the entry. You do not perform searches based on the attributes and attribute values contained in the DN itself. The sole purpose of the DN is to provide a unique name by which the entry is identified. There does not have to be any correspondence between the

attributes used for the entry's DN and the attributes actually used in the entry (although you should create this correspondence when creating directory entries to help you administer your directory tree).

Because there are no performance or administrative benefits to a branched directory tree, your best approach is to keep your directory tree as flat as possible. This does not mean that you should avoid hierarchy; some benefit can exist in logically dividing your entries so that you can tell by looking at the entry's DN what type of data it holds. Some common branch points directory services are now using include:

- `ou=People`
- `ou=Groups`
- `ou=Contracts`
- `ou=Employees`
- `ou=Contractors`

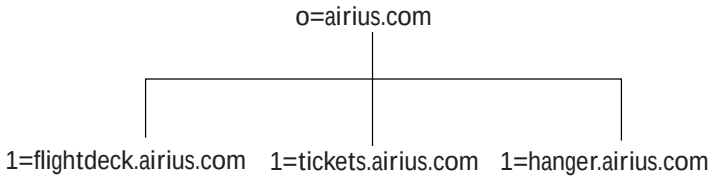
A directory tree organized along these lines may appear as follows:



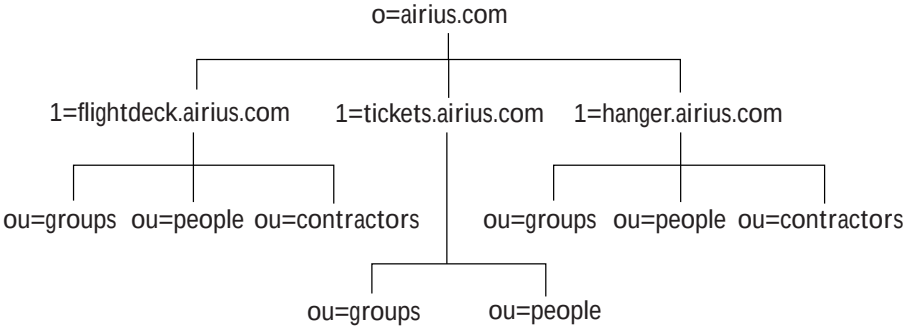
Branching to Support Replication or Referrals

You can organize your directory tree so that it corresponds to the network names in your enterprise. This approach is attractive because it is an extension of the strategy recommended for suffix names and because network names tend to be fairly stable entities. Further, this approach may be particularly attractive in the higher branches of your trees if your directory service is using replication or referrals to tie together different Directory Servers.

For example, if Airius has three primary networks known as `flightdeck.airius.com`, `tickets.airius.com`, and `hanger.airius.com`, then you might want to branch your directory tree as follows:



After your initial branching, you may want to further branch your directory tree along some of the divisions as noted in “Branching Strategies” on page 86. This is especially true if each Directory Server is maintaining directory entries specific to that location, such as might be the case for enterprises using subtree replication and local mastering of directory data or enterprises using smart referrals to tie together different organizational entities. For example:



Branching to Support Access-Control

In some respects, the reasons why you might branch to support access-control are similar to the reasons why you might branch to support replication: it is easier to group together similar entries and then administer them from a single branch point.

However, in the case of access-control, you may have some entries to which you want to grant special privileges based on the organization to which the entries belong. That is, you may want to grant a person write privileges to entries belonging to the Marketing organization. You can, of course, branch your directory tree to include a Marketing organizational tree. However, as stated in “Branching Your Directory Tree” on page 84, branches based on organizational divisions within your enterprise can lead to administrative problems when those divisional lines change.

Generally you can avoid branching for this reason by using the Netscape ACI filtered target mechanism. This mechanism allows you to apply an access-control rule to a directory entry based on some attribute-data pair included on the entry. That is, you can allow or deny access if an entry includes the attribute-data pair of:

```
ou: Marketing
```

At some point, however, you are likely to find that using search filters to support your ACL is cumbersome and difficult to manage. It is at this point that you must decide whether it is easier to branch along organizational lines, or if you would rather support a complex set of LDAP filters in your security policy.

Ultimately you will probably have to use some combination of the two mechanisms. Here are some guidelines that can make this task easier for you:

- Branch your tree to represent only the largest organizational subdivisions in your enterprise. That is, any such branch points should be limited to divisions (Corporate Information Services, Customer Support, Sales and Professional Services, and so forth). Make sure that any such divisions that you represent in your directory tree are as stable as possible; do not perform this kind of branching if your enterprise is fluid and is likely to frequently reorganize.
- Use functional or generic names rather than actual organizational names to represent your branch points. Names change and you do not want to have to change your directory tree every time your enterprise renames its

divisions. Instead, use generic names that represent the function of the organization (for example, use *Engineering* instead of *Widget Research and Development*).

- If you have multiple organizations that perform similar functions, try creating a single branch point to represent that function instead of branching based along divisional lines. For example, if you have multiple marketing organizations, each of which might be responsible for a specific product line, still only create a single Marketing subtree. All marketing entries should then belong to that tree.

- Once you have created your major directory tree branches, apply your filtered-target ACIs to the branch points of the trees to which they apply. That is, if you are granting write privilege to entries belonging to *Marketing Widget I*, then place the ACI on the branch point to the marketing tree. This does three things for you:

  1. Depending on what attributes you are basing your ACI on, it potentially helps to reduce the complexity of the search filters.

  2. It helps to improve server performance by reducing the total number of entries to which the filter can apply.

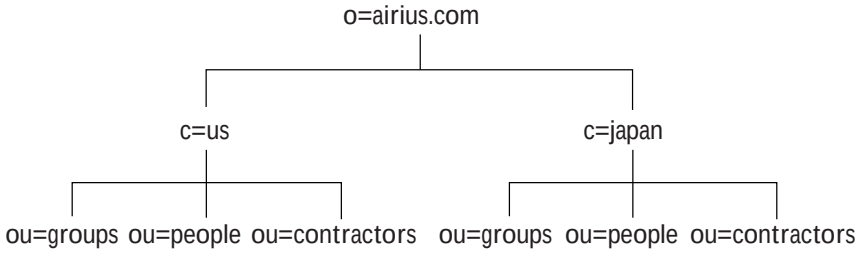
  3. It reduces any damage that may be caused in the event that the search filter is applied incorrectly and inappropriate entries are included in the scope of the ACI.

Branching to Support International Enterprises

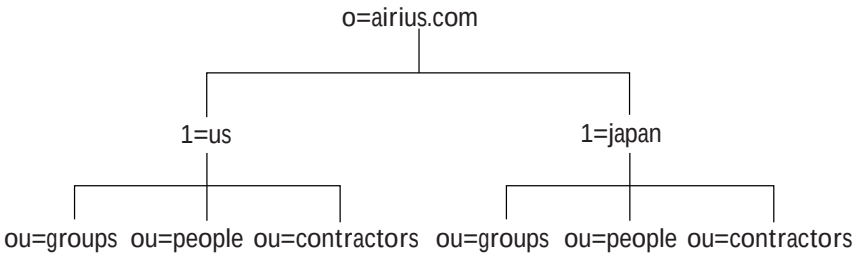
In “Selecting a Suffix” on page 80, you are advised to avoid rooting your directory tree in a country designator. This is especially true if your enterprise is international in scope.

To support an international enterprise, you should root your directory tree in your Internet domain name, then branch your tree for each country in which your enterprise has operations immediately below that root point. Because LDAP places no restrictions on the order in which DN attributes can appear in your DNs, you can use the `c` (`country`) attribute to represent each country branch.

For example:



However, some administrators feel that this is stylistically awkward, so instead you could use the `l` (`locality`) attribute to represent different countries:



Naming Person Entries

After you have branched your directory tree, you must name the person entries beneath your branch points. That is, you must decide what the relative distinguished name (RDN) will be for your person entries (the RDN is always the left-most attribute-data pair in any DN). Here again, the traditional choice is not necessarily the best choice.

Traditionally, distinguished names use the `commonName`, or `cn`, attribute to name their person entries. That is, an entry that represents a person named Babs Jensen might have the distinguished name of:

```
cn=Babs Jensen, o=Airius.com
```

While allowing you to instantly recognize the person represented by the entry, this format suffers in that most organizations usually include people with identical names. This quickly leads to a problem known as *DN name collisions*, that is, multiple entries with the same distinguished name. Remember that DNs must be unique.

One common way to avoid common name collisions is to add a unique identifier to the common name. For example:

```
cn=Babs Jensen 23, o=Airius.com
```

However, for very large directories, this can lead to awkward common names, such as:

```
cn=Bill Smith11022
```

A better approach is to identify your person entries with some attribute other than `cn`. There are two traditional possibilities available to you:

- `uid`—Use the `uid` (`userID`) attribute to specify some unique value associated with the person. Possibilities include a user login ID or an employee number.
- `mail`—Use the `mail` attribute to represent the person's email address. This option can lead to awkward DNs that include duplicate attribute values (for example: `mail=bjensen@airius.com, o=airius.com`), so you should use this option only if you cannot find some unique value that you can use with the uid attribute. Using the mail attribute may be appropriate if, for example, your enterprise does not assign employee numbers or user IDs for temporary or contract employees. The mail attribute in that case can be based on some email address used by the contractor on a permanent basis (for example: `ssarette@hotmail.com, o=airius.com`).

Whatever you decide to use for an attribute-data pair for person entry RDNs, you should make sure that they are unique, permanent values. Person entry RDNs should also be reasonably recognizable if possible. That is, `uid=bjensen, o=airius.com` is preferable to `uid=b12r56A, o=airius.com` because reasonably recognizable DNs make some directory tasks easier to manage. For example, when you manage groups you will probably have to manipulate directory entries based on their distinguished names. Also, some directory clients assume that the UID or CN attribute use a human-readable name.

Naming Non-Person Entries

Your directory will contain entries that represent many other things than people. Beyond branch point entries that might represent organizations, localities, states, countries, and so forth, your directory could also include entries that represent groups, devices, servers, network information, and other data that is unique to your enterprise.

For these types of entries, use the `commonName` (`cn`) attribute in the RDN if possible. Therefore, if you are naming a group entry, name it as follows:

```
cn=administrators, o=airius.com
```

The caveat is if you are naming an entry whose object class does not support the `commonName` attribute. Since you will want to include the RDN attribute and attribute value in the directory entry to aid in locating the entry, you should use an attribute for the RDN that is supported by the entry's object class (fortunately, there are not very many object classes in the standard schema where this is a problem).

Finally, if you decide to use a non-traditional attribute for your RDN, make sure that all of the applications and clients that will use your directory support the non-standard attribute. The Netscape 1.x Certificate Server in particular has restrictions on what DNs it will support in your directory. See the Certificate Server documentation for more information.

Chapter

7

Planning Replication

Every directory service should use replication to ensure data availability in the event that a server becomes unavailable. Therefore, replication is an important strategy that you should plan on using with your directory service.

In this chapter you will learn about replication, its concepts and uses. This chapter also provides advice on how and when to use replication. This chapter includes the following sections:

"Replication Overview" on page 96—This section briefly introduces replication, including the benefits that replication brings to your directory service. This section also includes a brief look at the supplier-consumer architecture used by Netscape's replication strategy.

"Replicating Directory Trees" on page 98—This section describes the kinds of replication that you can configure for your directory service. Topics include whole tree replication, subtree replication, cascading replication, and multiple subtree replication. This section also describes replication initialization, and directory configuration requirements for creating replication.

"Building a Highly Available Directory Service" on page 104—This section provides concepts and examples of how to use replication to build high availability into your directory service. DNS strategies, using replication for load balancing, and using replication for local availability are also discussed. Examples include how to load balance for a small site, a large site, for local data management, and for server traffic. A specific section on how to load balance for Netscape Messaging Server 3.0 is also provided.

Replication Overview

Replication is the mechanism by which directory data is automatically copied from one Directory Server to another. Using replication, you can copy everything from entire directory trees to individual directory entries between servers. Replication provides several important benefits to your directory service:

- Fault tolerance—By replicating directory trees to multiple servers, you can ensure your directory is available even if some hardware, software, or network problem prevents your directory clients from accessing a given Directory Server instance.
- Higher performance—By replicating directory entries to a location close to your users, you can vastly improve directory response times.
- Load balancing—By replicating your directory tree across servers, you can reduce the access load on any given machine, thereby improving server response time.
- Local data management—Replication allows you to own data locally and share it with other Directory Servers across your enterprise.

Before examining the issues behind replication usage, it is useful to understand Netscape's replication architecture.

To begin, every directory object must be mastered by one and only one Directory Server. This mastering Directory Server is called the *supplier server* because it supplies the object to other servers.

Servers that receive directory objects from supplier servers are called *consumer servers.*

Any given Directory Server can be both a supplier of directory objects as well as a consumer of objects supplied to it from other servers.

Supplier Servers

A supplier server is responsible for:

- Managing any requests for changes to the replicated directory data. That is, anytime a request to add, delete, or change an entry in a replicated tree is received by a consumer server, the request is referred to the supplier server where the request is actually performed.
- You can configure the supplier server to initiate replication, or you can configure your consumer server(s) to initiate the replication process. Either way, the supplier server is always responsible for tracking the changes made to the objects that it masters so that those changes can be replicated to consumer servers.

Consumer Servers

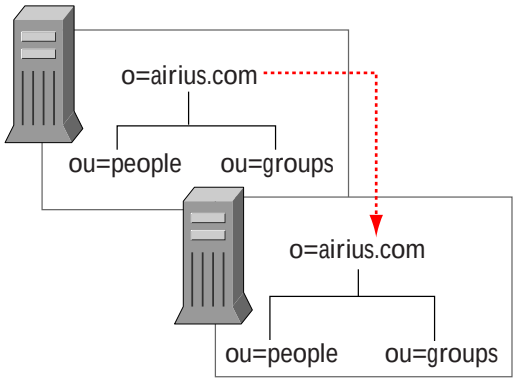
Consumer servers contain at least one directory entry that has been copied to it by a supplier server. Consumer servers can contain:

- The supplier server's entire directory tree
- A subsection, or a subtree, of the supplier server's directory tree

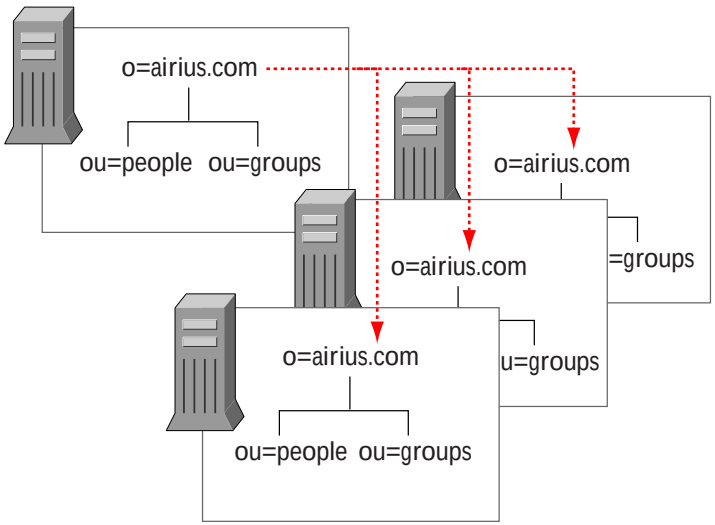
Only read operations occur on the consumer server; all other operations are handled on the supplier server. Anytime an LDAP client tries to modify entries (add, delete, or change any part of the entry) in a replicated tree, the consumer server automatically refers the LDAP client's request to the supplying server. For more information on referrals, see Chapter 8, “Planning Referrals.”

Replicating Directory Trees

In its most basic configuration, a supplier server replicates a directory tree to one or more consumer servers. In this configuration, all directory modifications occur on the supplier server, and the consumer server contains read-only data. The supplier server is responsible for managing (performing modifies to) all the directory data contained on the consumer server.



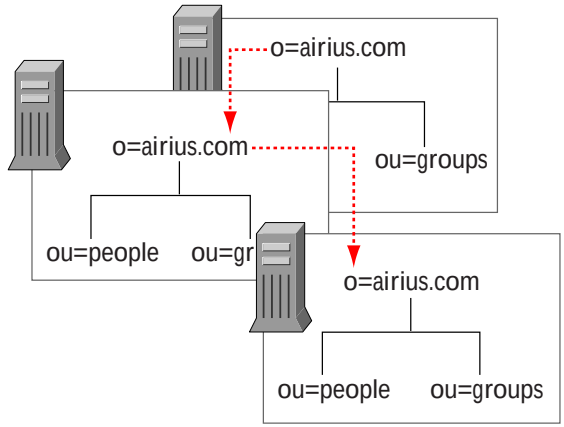
The supplier server can, of course, replicate its directory tree to more than one consumer server. The total number of consumer servers that a single supplier server can be manage is dependent upon the speed of your networks and the total number of entries changing on a daily basis. However, you can reasonably expect a supplier server to maintain several consumer servers.



Cascading Replicas

Cascading is a replication technique that involves a supplier server replicating directory data to a consumer server, and then that consumer server replicating the same data to yet another consumer or consumers.

This form of replication is useful, for example, if some network connections between various locations in your organization are better than others. For example, suppose you are mastering your directory data in Minneapolis, and you have consumer servers in Saint Cloud as well as Duluth. Suppose, too, that your network connection between Minneapolis and Saint Cloud is very good, but your network connection between Minneapolis and Duluth is of poor quality. Then, if your network between Saint Cloud and Duluth is of acceptable quality, you can use chaining to move directory data from Minneapolis to Saint Cloud to Duluth:

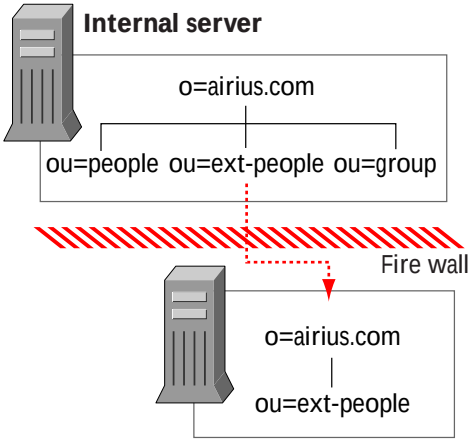


Subtree Replication

A supplier server does not have to replicate its entire directory tree to a consumer. Instead, the supplier server can replicate a portion of its tree to other servers. This form of replication is known as subtree replication.

One of the best uses of subtree replication is if you are enabling an extranet. In this case, you may want to master all of your directory data in a single server inside your firewall. However, your trading partners are likely to need access to some portion of that tree, which means it needs to be accessible outside your firewall.

Because it is best to avoid making internal, private information physically available on the Internet (even if your access-control mechanism secures that data from outside intrusion), you should create a special Directory Server that is available only outside your firewall. Then use subtree replication to copy directory data from your internal server to your external server. Only replicate that portion of your directory tree that you want your trading partners to be able to access.



Multiple Subtree Replication

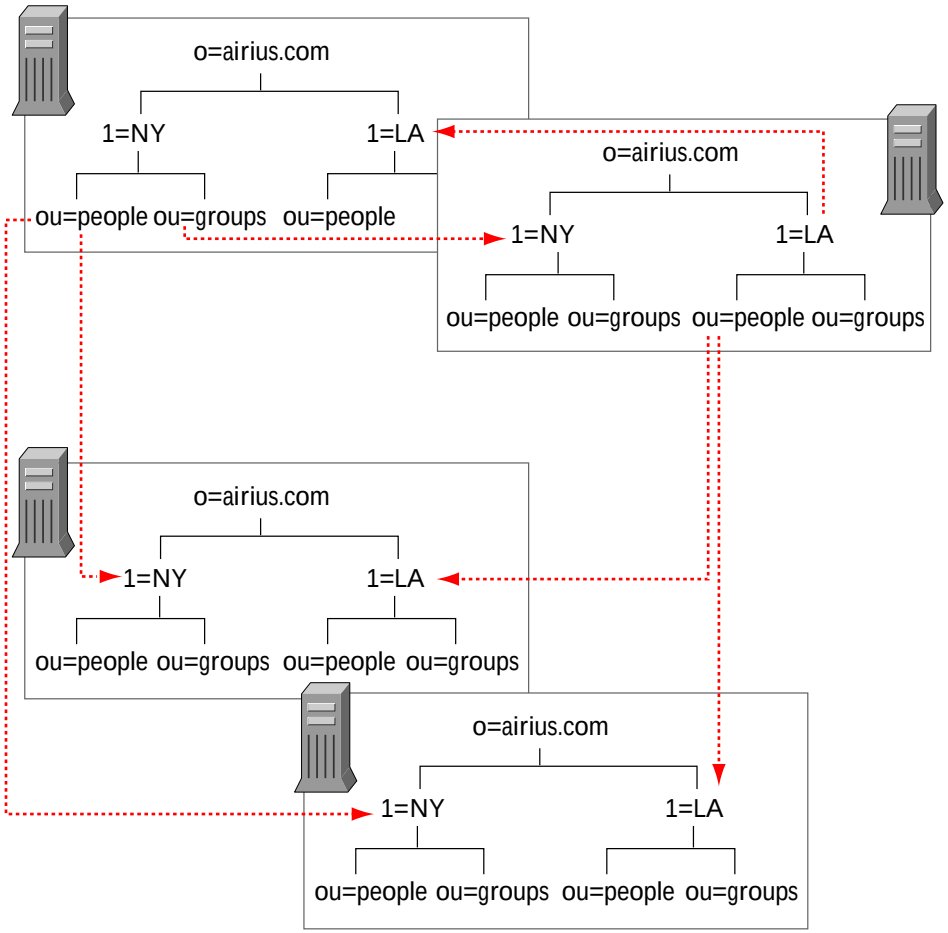
Large enterprises may find the need to replicate directory subtrees from multiple suppliers to a single consumer. For example, suppose your enterprise has main branch offices in New York and Los Angeles, with smaller satellite

offices in Chicago and Dallas. Management of your enterprise's directory tree is evenly split between New York and L.A. However, Chicago and Dallas need local copies of everything.

Then you should replicate subtrees between servers as follows:

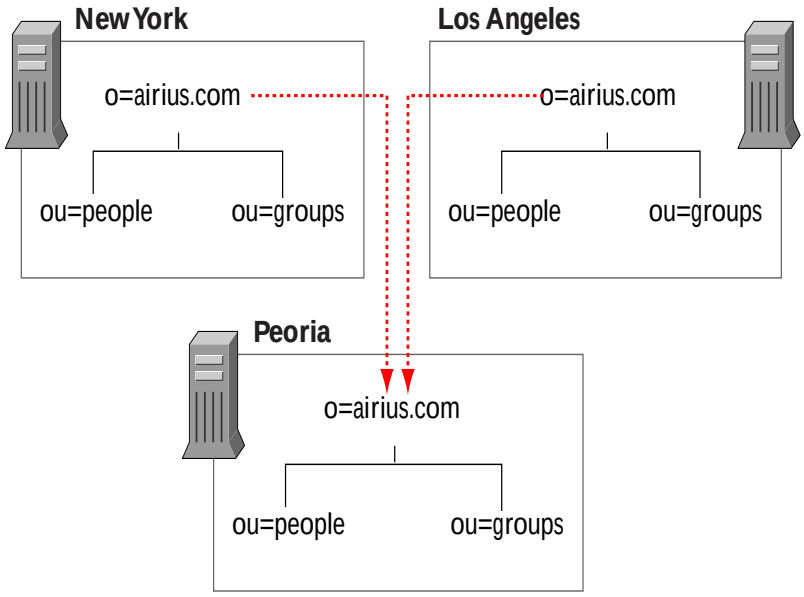
- New York supplies the subtree that it manages to Chicago, Dallas, and L.A.
- L.A. supplies the subtree that it manages to Chicago, Dallas, and New York.

Notice that in this arrangement New York and L.A. are both suppliers as well as consumers of directory data.



The one thing you cannot do with multiple subtree replication is have two different servers manage the same tree, and then replicate those subtrees to a single consumer.

For example:



Initiating Replication

Replication synchronization can be initiated by either the supplier or the consumer server. You choose which form of synchronization is used for each replication agreement. A *replication agreement* indicates what directory entries will be replicated, the servers between which the replication is occurring, and when the replication can occur.

If you are using supplier-initiated replication, it is the responsibility of the supplier server to determine when its consumer servers are to be updated.

If you are using consumer-initiated replication, it is the responsibility of the consumer server to determine when it wants to retrieve updates from the supplier server.

From an administrative perspective, there is no difference in managing these two types of synchronization. Both types of synchronization allow you to update the consumer server(s) on a set time interval. However, supplier-initiated replication allows you to configure the supplier server to update its consumer(s) the instant that an object is updated on the supplier, whereas consumer-initiated replication can only occur on a set interval. Therefore:

- If you want your consumer servers to be updated instantly, use supplier-initiated replication.
- If you are using a dial-up connection to update your consumer servers, use consumer-initiated replication.

Other than these conditions, there is no difference in using the two types of synchronization from either an administrative or performance perspective. In general, you should try these two types of synchronization in a lab environment and decide which one you are most comfortable with from a management perspective. Then, if possible, use that one type of synchronization across your enterprise so as to lower training costs for your IS personnel.

Directory Entries used to Support Replication

You must create special directory entries to support replication. The actual type of entry that you need is dependent upon the type of replication that you are using.

Supplier-Initiated Replication

For supplier-initiated replication, you configure each consumer server with a *supplier DN*. This special distinguished name is used by a supplier to bind to the consumer server and update the consumer's directory. If a directory client attempts to modify a replicated entry on the consumer server, and that client does not bind as the supplier DN, then the consumer server refers that request to the server that originally supplied the entry.

The supplier DN is not an actual entry in the directory tree. Instead, it is an "ethereal" or "virtual" entry. This entry is identified to the consumer server by a configuration parameter in the server's configuration file (see the *Netscape Directory Server Administrator's Guide* for more information).

Every consumer server must be configured with one and only one supplier DN. Since the supplier DN is a virtual entry, you do not have to set any access-control statements to allow a user binding as this entry to update the consumer server.

Consumer-Initiated Replication

For consumer-initiated replication, a directory entry must be created on the supplier that the consumer can use to bind to the supplier server. This entry must have read and search privileges for the tree that the consumer is retrieving from the supplier, as well as read and search privileges for the supplier server's change log. The change log is a special directory tree on the supplier server that the supplier uses to track changes to its directory tree.

For consumer-initiated replication, it is administratively easiest to create a single directory entry that every consumer server will use to perform the synchronization. However, you may want to create a unique entry for every consumer to use for this purpose. Doing so will allow you to see which exact server is accessing you supplier server when you view the access log. This can help you to resolve replication problems should any occur.

For more information on configuring replication, see the *Netscape Directory Server Administrator's Guide*.

Building a Highly Available Directory Service

Replication is a vital part of your directory strategy. This is because you are using your directory service to centralize data, most likely data that is mission critical, and thus you will have to use replication to ensure uninterrupted availability of your directory service.

You use replication to ensure high availability of your directory service in three basic ways:

- By duplicating data to backup servers
- By moving data to local servers for faster and more reliable access
- By duplicating data to multiple servers for load balancing purposes

Each of the following sections describe these strategies in detail.

Using Replication for High Availability

Use replication to avoid a situation in which the loss of a single server causes your directory service to become unavailable. At a minimum you should replicate the local directory tree to at least one backup server.

LDAP clients usually can be configured to search only one LDAP server. That is, unless you have written a custom client to rotate through LDAP servers located at different DNS hostnames, you can only configure your LDAP client to look at a single DNS hostname for a Directory Server. Therefore, you will likely need to use either DNS round robins or network sorts to provide fail-over to your backup Directory Servers.

DNS round robins are a feature available in most DNS servers that allows you to configure multiple IP addresses for a single DNS hostname. The DNS server then rotates the order of the IP addresses that it returns as the result of a DNS query. Therefore, if you configure DNS lookups of hostname `directory.airius.com` to return the IP addresses `123.456.789.1`, `123.456.789.2`, and `123.456.789.3`, the DNS server will return IP addresses in the following orders:

1st Lookup Results	2nd Lookup Results	3rd Lookup Results	4th Lookup Results
123.456.789.1	123.456.789.2	123.456.789.3	123.456.789.1
123.456.789.2	123.456.789.3	123.456.789.2	123.456.789.2
123.456.789.3	123.456.789.1	123.456.789.1	123.456.789.3

Since DNS clients try the IP addresses in the order that they are returned, your LDAP clients will only have to time out once every third request if the machine at one of the three IP addresses becomes available. If the server will be unavailable for a long time, its IP address can be removed from the DNS round robin so as to avoid repeated time out situations.

The problem with DNS round robins is that they make no attempt to prioritize DNS results based on network location. It is almost always more desirable to try hosts on the local network before trying hosts that are farther away in your network topology.

To solve this problem, some more recent versions of DNS allow network sorts. A network sort allows you to identify a set of IP addresses that are a best choice and a set of IP addresses that are a second best choice. The DNS server will always return the best choice IP addresses at the top of the list with the second best choice at the bottom. The round robin then rotates the top and bottom choices within their groupings. Thus, with network sort, DNS queries would return IP addresses in the following order:

	1st Lookup	2nd Lookup	3rd Lookup	4th Lookup
Best choice	123.456.789.1	123.456.789.2	123.456.789.1	123.456.789.2
	123.456.789.2	123.456.789.1	123.456.789.2	123.456.789.1
	123.456.789.3	123.456.789.4	123.456.789.5	123.456.789.3
Next choice	123.456.789.4	123.456.789.5	123.456.789.3	123.456.789.4
	123.456.789.5	123.456.789.3	123.456.789.4	123.456.789.5

Round robins and network sorts are a viable strategy only for read-only operations. You cannot mirror supplier servers on multiple machines and use DNS round robins to provide automatic fail-over. If your supplier server becomes unavailable, you must manually convert a consumer server to the supplier server and then reinitialize the remainder of your consumer servers.

For information on setting up and using DNS round robins or network sorts, see your DNS documentation.

Using Replication for Load Balancing

On average, a Directory Server running on a reasonably fast machine can be expected to handle around 800 search requests per second and thousands of simultaneous directory connections. However, there are a great many factors that can impact server performance, including:

- The amount of RAM available to your server. For best performance you should provide enough RAM that your entire database resides in memory (although this is an unreasonable expectation if your database is very large). The more disk accesses that your server must perform, the slower it will run.
- The number of entries in your database.
- The number and types of indexes that you use on your server. Indexes are used by the Directory Server to rapidly respond to search requests. There are quite a few different types of indexes that you can use with your server, and their usage is important to tuning your server's performance. For detailed information on Directory Server indexes, see the *Netscape Directory Server Administrator's Guide.*
- The amount of write activities performed on your server. Directory writes are significantly slower than directory searches because they require disk accesses and may require index creation. Therefore, the more writes that your server must perform, the slower your overall search performance will be.
- The speed of your networks.
- The number of access-control statements included in your directory. The more ACIs your Directory Server must examine before responding to a search request, the slower it will be.

You can use replication to balance the load on your directory service by:

- Spreading your user's search activities across several servers
- Dedicating servers to read-only activities (writes occur only the supplier server)
- Dedicating special servers to specific tasks, such as supporting mail server activities

Load Balancing the Server

As the load on your directory service grows, you will have to increase computing power to service the increased traffic. Your need to add hardware to handle increased directory access depends on quite a few factors, including CPU speed and the amount of RAM available to the server. In some cases you may be able to simply add additional RAM to handle an increased load. However, at some point you will have to add a new server entirely.

In terms of mapping the number of supported search requests per second to real-world loads, consider the following:

- Each time a user performs a directory lookup represents a single search request. A good rule of thumb is to assume that every user will perform an average of 10 search requests per day. Therefore, if you have 10,000 people using your directory service, you are looking at approximately 100,000 search requests per day, or about 1.15 search requests per second over a 24-hour period. If your enterprise is contained within just a few time zones, then a more likely figure for the same number of users might be 3.47 search requests per second over an 8-hour period.

- Each time Netscape Messaging Server 3.0 handles a single piece of mail, it performs 5 search requests. On average, a SuiteSpot 3.5 Messaging Server can handle around 25 mails per second. Therefore, each messaging server using your directory represents a maximum load of 125 reads per second.

- Other Netscape servers access the directory only if access-control is turned on for the server. Each time a user authenticates to the server (such as a Collabra Server), 2 searches are performed against the directory (once to obtain the user's DN, and a second time to perform the actual authentication).

Load Balancing the Network

One of the more important reasons to replicate directory data is to load balance your network. When possible, you should move data to servers that can be accessed using a reasonably fast and reliable network connection. The most important considerations are the speed and reliability of the network connection between your server and your directory users.

In terms of network load, directory entries generally average around 1 KB in size. Therefore, every directory lookup adds about 1 KB to your network load. If your directory users perform around 10 directory lookups per day, then for

every directory user you will see an increased network load of around 10,000 bytes per day. Given a slow, heavily loaded, or unreliable WAN, you may need to replicate your directory tree to a local server.

You must carefully consider whether the benefit of locally available data is worth the cost of the increased network load due to replication. For example, if you are replicating an entire directory tree to a remote site, you are potentially adding a large strain on your network in comparison to the traffic caused by your users' directory lookups. This is especially true if your directory tree is changing frequently, yet you only have a few users at the remote site performing a few directory lookups per day.

For example, consider that your directory tree can easily include in excess of 1,000,000 entries and that it is not unusual for approximately 10% of those entries to change every day. If your average directory entry is only 1 KB in size, this means you could be increasing your network load by 100 MB/day. However, if your remote site only has a few employees, say 100, and they are performing an average of 10 directory lookups a day, then the network load caused by their directory access is only 1 MB per day.

Given the difference in loads caused by replication versus that caused by normal directory usage, you may decide that replication for network load-balancing purposes is not desirable. On the other hand, you may find that the benefits of locally available directory data far outweigh any considerations you may have regarding network loads.

Using Replication for Local Availability

There are several reasons why you may want to replicate for local availability. They include:

- You are mastering data locally. This is an important strategy for large, multinational enterprises that need to maintain directory information of interest only to the employees in a specific country. Local data mastering is also important to any enterprise where interoffice politics dictates that data be controlled at a divisional or organizational level.
- You are using unreliable or intermittently available network connections. In this case you may want to replicate directory data when the state of the network supports the traffic. These kinds of conditions can occur if you are using unreliable WANs, such as often occurs in international networks.

- Your networks periodically experience extremely heavy loads that may cause the performance of your directory service to be severely reduced. For example, enterprises with aging networks may experience these conditions during normal business hours.

Your need to replicate for local availability is determined by the quality of your network as well as the activities of your site. In addition, you should carefully consider the nature of the data contained in your directory and the consequences to your enterprise in the event that the data becomes temporarily unavailable. The more mission critical this data is, the less tolerant you can be of outages caused by poor network connections.

Determining Your Replication Strategy

Although there is no way to predict the actual amount of activity that your directory service will experience, some of the factors that will affect directory activity are:

- The number and size of the entries you contain in your directory service
- The number of users you have accessing your directory service, and their habits with regard to directory lookups
- The number of applications you have using your directory, and their activities with regards to the directory
- The quality of your LANs and your WANs, including the stability of your network connections and the amount of bandwidth available on them

To determine your replication strategy, start by performing a survey of your enterprise's networks. Also examine the physical locations of your users and look at the quality of the WANs connecting those sites.

As a part of this survey, you should find out how many users are at each site and estimate how often directory lookups will be occurring from those locations. For example, a site that manages HR databases or financial information is likely to put a heavier load on your directory service than a site containing engineering staff that uses the directory service for simple telephone book purposes.

Once you have a good understanding of your enterprise's infrastructure, consider the performance numbers discussed earlier in this chapter. As you develop an understanding of your replication needs, make sure you include the following strategies in your replication plans:

- Replicate locally to any site where the quality of the WAN may cause remote Directory Servers to become unavailable. This is especially important if the nature of your directory data is vital to daily operations.
- If you have a remote site that is responsible for managing some aspect of your directory tree, make sure that site has a local server that is used to master the data.
- At each site where you place a Directory Server, replicate at least once for fault tolerance purposes. Some directory architects argue that you should replicate three times per physical location for maximum data reliability. The extent to which you replicate for fault tolerance purposes is up to you, but you should base this decision in part on the quality of the hardware and networks that are supporting your directory service. Essentially, unreliable hardware requires more backup servers.

Once you have a basic understanding of your replication strategy, you can start deploying your directory service. This is a case where deploying your service out in controlled stages will pay large dividends (see **"Deployment Advice" on page 25**). By placing your directory service into production in stages, you can get a better sense of the loads that your enterprise places on your directory service. Unless you have an existing directory service to base your load analysis on, be prepared to alter your directory service as you develop a better understanding on how your directory is used.

Remember throughout this design phase that the Netscape Directory Server's replication feature is extremely flexible. You will find it very easy to scale or alter your directory topology as the loads on your Directory Server change. Therefore, if you find that you have misjudged the load placed on your service, you can easily add, remove, or move servers to better handle your enterprise's actual requirements.

Example: Load Balancing a Small Site

Suppose your entire enterprise is contained within a single building. This building has a very fast (100 MB/second) and lightly used network. The network is very stable and you are reasonably confident of the reliability of your server hardware and OS platforms. Also, you are sure that a single server's performance will easily handle your site's load.

In this case, you should replicate at least once to ensure availability in the event your primary server is shut down for maintenance or hardware upgrades. Also, set up a DNS round robin to improve LDAP connection performance in the event that one of your Directory Servers becomes unavailable.

Example: Load Balancing a Large Site

Suppose your entire enterprise is contained within two buildings. Each building has a very fast (100 MB/second) and lightly used network. The network is very stable and you are reasonably confident of the reliability of your server hardware and OS platforms. Also, you are sure that a single server's performance will easily handle the load placed on a server within each building.

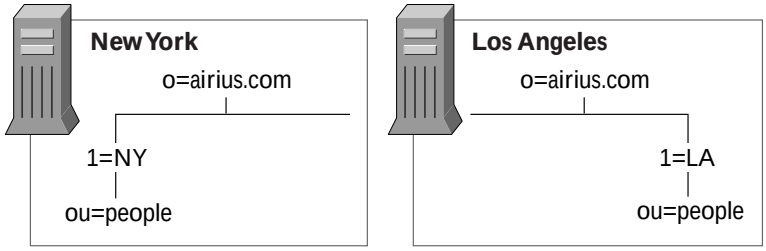
Also assume that you have slow (ISDN) connections between the buildings, and that this connection is very busy during normal business hours.

In this case, you should do the following:

- Choose a single server in one of the two buildings to master your directory data. This server should be placed in the building that contains the largest number of people responsible for mastering directory data. Call this Building A.
- Replicate at least once within Building A for high availability of directory data.
- Create two replicas in the second building (Building B).
- If there is no requirement for tight consistency between the supplier and consumer server, schedule replication so that it occurs only during off peak hours.

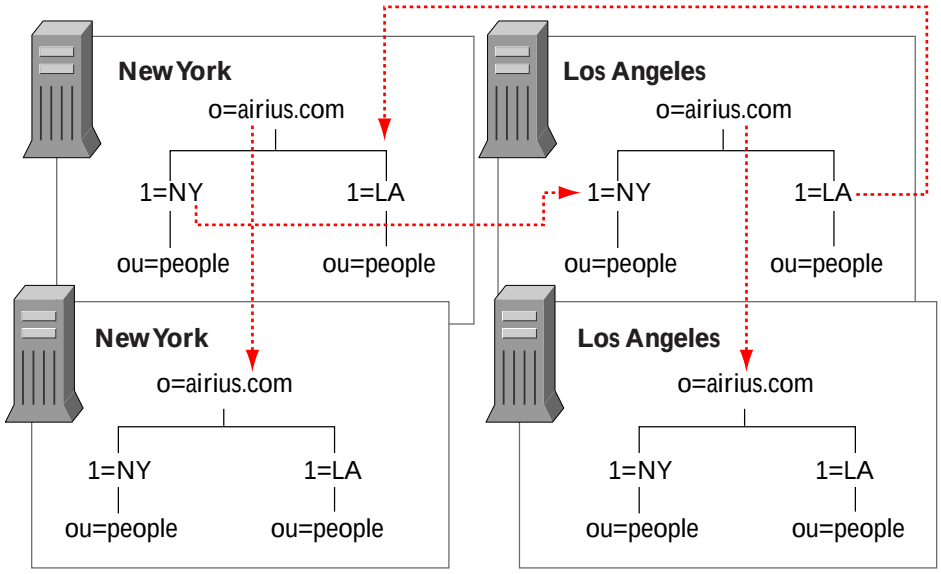
Example: Load Balancing for Local Management

Suppose you have an enterprise with offices in two cities. Each office has specific subtrees that they are responsible for managing as follows:



Each office contains a high-speed network, but you are using a dial-up connection to network between the two cities. Do the following:

- Select one server in each office to be the master server for the locally managed data. Replicate locally managed data from that server to the corresponding master server in the remote office.
- The master server in each office should replicate its entire directory tree (including data supplied from the remote office) to at least one local consumer server to ensure availability of the directory data.



Example: Load Balancing for Server Traffic

Suppose that your directory service must include 1,500,000 entries in support of 1,000,000 users, each of which is performing 10 directory lookups a day. Also assume that you are using Netscape Messaging Server 3.0 which is handling 25,000,000 mail messages a day. Since the 3.0 messaging server performs 5 directory lookups for every mail message that it handles, you can expect to see 125,000,000 directory lookups per day just as a result of mail. Your total combined traffic is therefore 135,000,000 directory lookups per day.

Assuming an 8-hour business day, and that your 1,000,000 directory users are clustered in 4 time zones, your business day (or peak usage) across 4 time zones is 12 hours long. Therefore you must support 135,000,000 directory lookups in a 12-hour day. This equates to 3,125 lookups per second (135,000,000 / (60*60*12)).

That is:

1,000,000 users	@ 10 lookups per user=	10,000,000 reads/day
25,000,000 messages	@ 5 lookups per message =	125,000,000 reads/day
	Total reads/day =	135,000,000
12-hour day includes 43,200 seconds	Total reads/second =	3,125

Now, assume that you are using a combination of CPU and RAM with your Directory Servers that allows you to support 500 reads per second. Simple division indicates that you need at least 6-7 Directory Servers to support this load. However, for enterprises with 1,000,000 directory users, you should add additional Directory Servers for local availability purposes.

You could, therefore, replicate as follows:

- Place a single Directory Server in one city to handle all write traffic. This assumes that you want a single point of control for all directory data.
- Use that one server to replicate to a single consumer server. This reduces the amount of work that your master server has to perform, thereby freeing it up to handle write requests. Call this one consumer server the *replication master.*
- Use the replication master to replicate to individual sites throughout the enterprise. This replication will help load balance your servers, and your WANs, as well as to ensure high availability of directory data. Assume that you want to replicate to four sites around the country. You then have four consumers of the replication master.
- At each site, replicate at least once to ensure high availability. Use DNS sort to ensure that local users always find a local Directory Server against which they can perform directory lookups.

Example: Replicating for Messaging 3.0

Once you start looking closely at server performance numbers, you will find that the Directory Server is usually not the part of your intranet that causes performance problems. Generally, server for server, you will need more instances of other types of Netscape servers to handle your intranet load than you will need Directory Servers.

For example, suppose your Netscape Directory Server is running on a combination of CPU and memory that allows it to handle 600 to 700 search requests per second. Conversely, a 3.0 Netscape Messaging Server can handle around 25 messaging requests per second. Therefore, for any appreciable intranet population, you will need many more Messaging Servers than Directory Servers.

The question therefore is: how should you best deploy Directory Servers to support a large population of messaging servers? While you will not need to replicate for directory load balancing, you will want to replicate for high availability.

Depending on the quality of your hardware and networks and on the size of your user population, there are two approaches that you may want to take. The first is to replicate only that set of directory data to a Directory Server instance running on the same physical host as the messaging server instance is running. This has the advantage of providing the highest availability of directory data possible to the messaging server. However, it also has the disadvantage of being slightly harder to manage because every time you want to move a messaging account from one messaging server to another, you have to revisit your directory replication scheme. Therefore, for large user populations that may require constant movement of users from mail hosts to mail hosts, you should avoid this strategy.

A second strategy is to replicate to a central Directory Server residing on the same network as your messaging servers. The Netscape Directory Server can support roughly four or five 3.0 Messaging Servers at a time (each messaging server can process around 25 mails per second, and each mail requires 5 directory lookups). Consequently, place 1 Directory Server instance for every 4 or 5 messaging servers that you are using. This Directory Server should be on the same physical network as your messaging servers, and as always you should replicate this local server at least once for availability purposes.

Depending on the number of messaging servers that you are supporting with your Directory Server, you may want to dedicate the Directory Server to handling only messaging lookups. This is especially true if you are replicating to a Directory Server instance running on the same physical host as the messaging server. When you dedicate a Directory Server to support only messaging lookups, do the following:

- Replicate only those directory entries that the messaging server cares about. That is, replicate only those directory entries that represent mail accounts within your enterprise. You can do this by placing all the entries that the messaging server cares about into a specific subdirectory.

- Limit the amount of indexing that you are performing in your Directory Server. Indexes are described in detail in the *Netscape Directory Server Administrator's Guide*. However, briefly, indexing allows your Directory Server to rapidly find entries in your directory database. You can control the type of indexing your server performs (substring, equivalence, presence, and so on) for each attribute that you want to index. The more indexes you support on your server, the more RAM requirements your server will have and the more of a performance hit your server will take whenever directory entries are changed. Thus, you should to limit your indexes to only those attributes that the messaging server routinely accesses. Specifically, you should index as follows for messaging server usage:

```
index   cn pres,eq,sub
index   mail pres,eq,sub
index   mailAlternateAddress eq
index   mailHost eq
index   member eq
index   sn pres,eq
index   aci pres
index   uid pres,eq,sub
index   uniquemember eq
```

- Sites with a very large messaging solution may want to dedicate Directory Servers to clusters of 3 to 5 messaging servers. This reduces some of the problems caused by moving people between messaging servers, as long as you design your messaging clusters so that most people will be moved between servers within the same cluster.

Chapter

8

Planning Referrals

Referrals are an important mechanism that allow you to scale your directory service to extremely large proportions. Not every directory service will use referrals, but you should at least consider them when designing your directory service.

In this chapter you will learn about referrals. This chapter includes:

“Referral Overview” on page 120—This section introduces referrals and describes how and when the mechanism is used to redirect directory client requests.

“Smart Referrals” on page 120—This section introduces smart referrals, how they can be used, and when you should use them. This section also provides some brief advice and tips on how you should use smart referrals.

“LDAP Client Referral Handling” on page 125—This section describes how LDAP clients behave when a referral is returned to them in response to a directory query.

Referral Overview

Referrals are a redirection mechanism that your directory service returns when a directory client requests a directory entry that does not exist on the local server. The Directory Server determines whether a referral should be returned by comparing the DN of the requested directory object against the directory suffixes supported by the local server. If the DN does not match the supported suffixes, the Directory Server will return a referral.

For example, if a directory client requests the directory entry `uid=bjensen, ou=people, o=airius.com` and the local machine is configured to manage only entries stored under the `o=netscape.com` tree, then a referral is returned to the client.

Referrals are not returned if:

- The client attempts to read an entry that could exist in the directory tree (the requested entry's DN suffix matches the directory DN suffix), but which simply does not exist or cannot be returned because of insufficient authority.
- The client attempts to modify (change or delete) an entry in the directory tree that does not exist or cannot be accessed because of insufficient authority.

For each of the above conditions, the Directory Server returns either an error or no response at all.

Smart Referrals

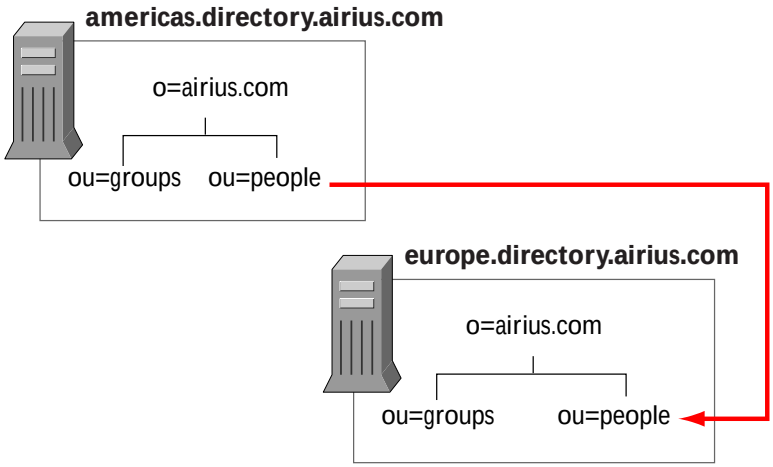
Netscape Directory Server version 3.0 and higher allows you to configure your Directory Server to use smart referrals. Smart referrals are a part of the LDAPv3 protocol.

Essentially, smart referrals allow you to map a directory entry or directory tree to a specific LDAP URL. This allows you to refer LDAP requests to:

- The same name space contained on a different server
- Different name spaces on the local server
- Different name spaces on the same server

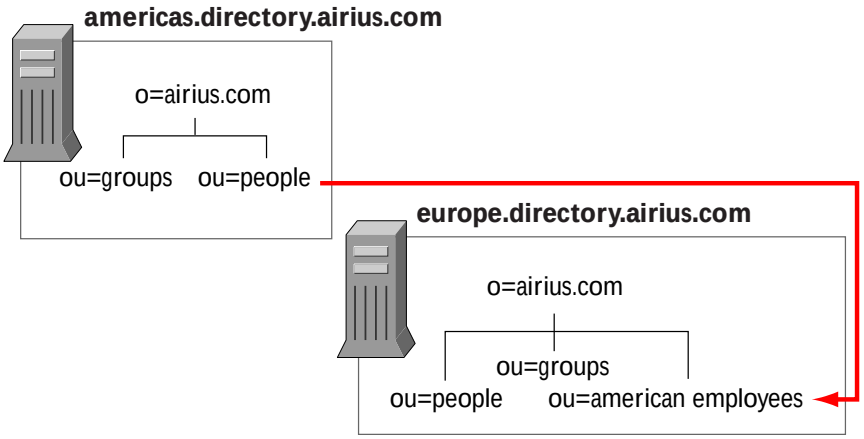
For example, if you have the directory branch point `ou=people, o=airius.com` and you want to redirect all requests for that branch to the server `directeurope.directory.airius.com`, then you can specify a smart referral on the branch point entry such as:

```
ldap://europe.directory.airius.com:389/ou=people, o=airius.com
```



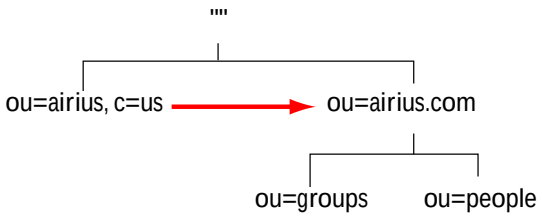
You can use the same mechanism to redirect queries to a different server that uses a different name space. For example, you can redirect requests to `ou=american employees, o=airius` corporation on the europe Directory Server:

```
ldap://europe.directory.airius.com:389/ou=american employees, o=airius
corporation
```



Finally, if you are serving multiple suffixes on the same machine, you can redirect queries from one name space (or directory tree) to another name space served on the same tree. That is, if you want to redirect all queries on the local machine for `o=airius, c=us` to `o=airius.com`, then you would put the following smart referral on the `o=airius, c=us` entry:

```
ldap:///o=airius.com
```



The third slash in this LDAP URL indicates that the URL is pointing to the local machine.

For information on LDAP URLS and on how to include smart URLs on Netscape Directory Server entries, see the *Netscape Directory Server Administrator's Guide.*

Smart Referral Usages

Because of their extreme flexibility, smart referrals allow you to scale your directory across multiple servers without physically containing those directory entries on each server in your enterprise. All that is needed is a referral from one entry in your local directory tree to an entry on some other server in your enterprise.

This referral mechanism is particularly useful for the following conditions:

- Enabling extranets—Smart referrals allow you to redirect directory lookup requests for your trading partners to your partner's public LDAP directory service. Of course, this has to be coordinated with your trading partners, but the potential still exists to allow users of your directory to easily find members of your partner's enterprise. A common usage of this is to allow people involved in joint, corporate ventures to find one another without having to replicate directory entries from one enterprise to another.

- Scaling—For extremely large enterprises, such as some national ISPs, referrals allow you to scale your directory service beyond the millions of directory entries that can be stored on a single Directory Server.

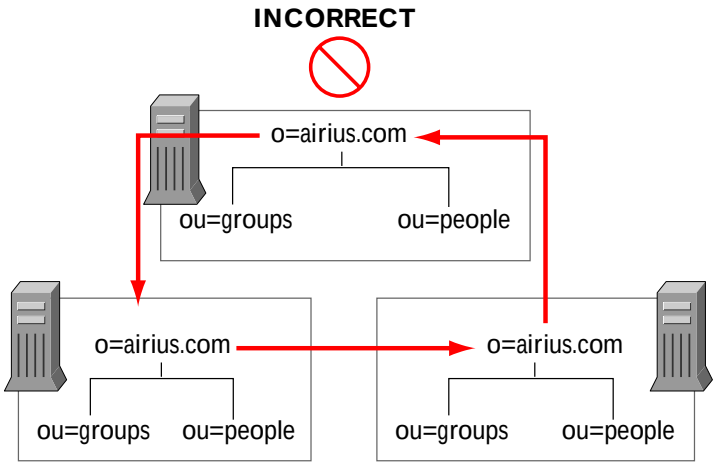
- Load balancing—In some situations, you may feel that replicating some directory entries to a specific Directory Server is inappropriate due to bandwidth or security constraints. However, you may also find that users of that remote directory service have an infrequent, but legitimate, need to access these same directory entries. Smart referrals allow those remote employees to locate the non-replicated entries simply by performing searches on their own local server.

How You Should Use Smart Referrals

Smart referrals are easy to use, but there are a few things to consider when using them:

- As always, keep it simple. Deploying your directory service in such a way that you have a tangled web of referrals will cause you administrative headaches. Also, over usage of smart referrals can lead to circular referral patterns. A circular referral is a situation in which a referral points to an LDAP URL, which in turn points to another LDAP URL, and so on until a referral somewhere in the chain points back to the original server.

For example:



- Limit your referral usage to handle redirection at a suffix level of a directory tree, or at least at a major branching point of your directory tree. One of the interesting aspects of smart referrals is that you can redirect lookup requests for leaf (non-branch) entries to other machines and completely different DNs entirely. As a result, you may be tempted to use smart referrals as an aliasing mechanism. However, this can quickly lead to a directory structure that is hard to manage and difficult to control, especially with regard to your security policy. By limiting referrals to the suffix or major branch points of your directory tree, you can limit the number of referrals that you have to manage, which in turn will significantly reduce your directory's administrative overhead.

- Access control does not cross referral boundaries. That is, if a type of access is allowed by one server, that same access is not automatically allowed by the referral server. Even if the server on which the request originated allows access to an entry, if the client is sent to another server using a smart referral then the server that actually contains the data must allow the directory access or the client's request cannot be fulfilled.

LDAP Client Referral Handling

How a referral is handled is determined by the LDAP client that you are using. Some clients will automatically retry the operation on the server to which the client has been referred. Other clients will simply return the referral information to you. For any Netscape-provided LDAP client (such as the client command-line utilities or the Directory Server gateway), the referral is automatically followed. If you supply bind credentials on the initial directory request, then those same bind credentials are used to access the server to which the client is referred.

One limitation that is built into most clients that can follow referrals is the number of referrals, or *hops*, that the client will perform. Most clients are coded to limit the number of hops that they will perform to reduce the amount of time spent trying to serve a directory lookup request. This also helps to eliminate hung processes caused by circular referral patterns.

Netscape Client Referral Handling

When a Netscape LDAP client is returned a referral, by default the client automatically reformats the original LDAP request to fit the boundaries set by the referral, and then the client reissues the request. Thus, if you search for an entry starting at the search base of `o=airius.com`, and on that server, the branch `ou=people, o=airius.com` is a smart referral to:

```
ldap://directory.europe.airius.com/ou=people, l=europe, o=airius.com
```

then the client will automatically attempt to search the directory and tree represented by that URL. In this case, the search base is modified to fit the referral DN. The new search base will therefore become `ou=people, l=europe, o=airius.com` because that is the value included on the LDAP URL.

The success of this operation will depend on the permissions set at the remote directory, as well as the availability of the remote directory.

In similar fashion, if you attempt to modify an entry and you are referred to another LDAP URL, then the client will attempt to reformat the modification request to fit the boundaries set by the referral. For example, suppose you attempt the entry `uid=bjensen, ou=people, o=airius.com`. Further suppose that `ou=people, o=airius.com` contains a smart referral to:

```
ldap://directory.europe.airius.com/ou=people, o=airius corporation
```

Both the server and the name space have changed (`o=airius.com` is now `o=airius corporation`). In this case, the server reformats the original modification request to fit the new name space and reissues the request on the new server. Assuming that the client has sufficient privileges to perform the modification, the operation is performed without the user ever knowing that the activity occurred on a remote server.

This reformatting operation has its limitations. For example, suppose you were issuing the modification request for `uid=bjensen`, and the smart referral pointed to an LDAP URL such as:

```
ldap://directory.europe.airius.com/o=airius corporation
```

Notice that not only have the server and the name space changed, but that the referral is no longer pointing at the `ou=people` subtree. If `uid=bjensen` entry is contained at the root point of the remote server, then the request will be handled without any complications. However, if the entry is contained in some subtree on the remote server (such as `ou=people`) then the server will not be able to manage the request because it will not be able to find the `uid=bjensen` entry.

Therefore, to support directory modifications with smart referrals, make sure that at least one of the following two conditions are true:

- All the relevant servers maintain parallel directory structures from the suffix down.
- A smart referral from a subtree in one server, points to the appropriate branch point in a remote server. That is, if you have a smart referral from the `ou=people` tree on one server, and the remote server stores people entries in the `ou=full time employees` tree, then the smart referral should be (for example):

  ```
  ldap://directory.europe.airius.com/ou=US employees,
  o=airius corporation
  ```

Chapter

9

Directory Design Examples

This chapter contains the following examples of the possible ways to organize your directory tree based on the size and nature of your enterprise:

"A Small Organization" on page 128—This example provides a detailed example on how you should design your directory tree for a small organization with simple data needs. This section forms the basis for all other types of directory tree design.

"A State Government" on page 131—This example shows you how you can design your directory tree for a state government, or any other organization comprised of a loose collection of independently operating organizations.

"An International Corporation" on page 133—This example shows you how you can design your directory tree to support a large, multinational organization.

"Enabling the Extranet" on page 139—This example shows you how you should publish directory information to support the extranet.

A Small Organization

Suppose your enterprise is a small organization consisting of a few hundred to a few thousand employees, or an individual division within a larger business. Also suppose that you have no plans to integrate your directory service with an X.500 directory.

The only purpose of this directory is to maintain user and group information in support of an Netscape server-based intranet that you are deploying through your organization. You want most of your user and group information to be centrally managed by a group of directory administrators. However, you also want email information to be managed by a separate group of mail administrators.

Finally, your organization consists of three organizational units: Engineering, Marketing, and Accounting.

Your directory tree should appear as follows:

- Root your directory tree in an Internet domain name or, if no such name is available, in the actual name of your organization. Do not root your directory tree in a country designator. Use the organization attribute (`o`) to root your directory. For example: `o=airius.com`
- Create two primary branch points in your directory tree: `ou=people` and `ou=groups`.
- Under `ou=people`, create all of your people entries. Use the `person` or `inetOrgPerson` object class for these entries. Also, since Netscape servers requires you to populate all user entries with a unique `uid` attribute value, you should use this value to uniquely identify each entry's DN.

  Also for each person entry, create and populate an `organizationalUnit` (`ou`) attribute with the organization that the person belongs to (Engineering, Marketing, or Accounting).

  You can, if you wish, also populate each person entry with a second `organization` (`o`) attribute that represents your directory root point, but this is strictly optional.

- In your group subdirectory, you need to create at least two groups. The first is used by your directory administrators to manage your directory contents. For example, use a group called `cn=administrators, o=<your suffix>`.

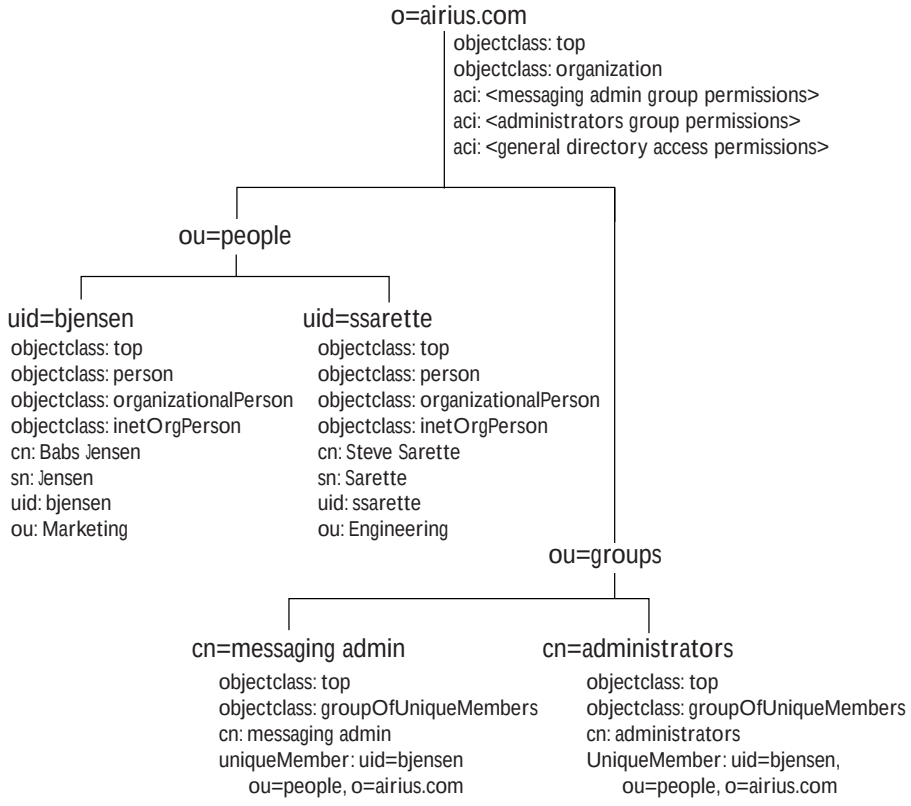
  The second group that you need to create is used by your mail administrators to manage mail accounts. This group should correspond to the administrators group used by the messaging server's administration server. Make sure the group that you configure for the messaging server is different than the group you create for the Directory Server. For example: `cn=messaging admin, ou=groups, o=<your suffix>`.

- At the root point of your directory tree, create your ACIs that allow your two groups the appropriate directory permissions. Your directory administrators group needs full access to the directory. Your messaging administrators group needs write and delete access to the `mailRecipient` and `mailGroup` object classes, the attributes contained on those object classes, as well as the `mail` attribute. You may also want to grant the messaging administrators group write, delete, and add permissions to the group subdirectory for creation of mail groups.

- At the root point of your directory tree, define some general access-control privileges, such as anonymous access for read, search, and compare, or authenticated read, search, and compare access. The Netscape Communicator Address Book feature requires anonymous access read, search, and compare access.

Remember that although this is a simple directory service that is containing relatively few entries, you should still replicate your entire directory tree at least once to ensure high availability of your directory service.

The following shows a sample directory tree designed as described above:



A State Government

If you are designing a directory tree for a state government, you most likely will want to provide room for city and county governments, for state and community colleges, and for various state and local agencies to participate in your directory tree.

Also, just as an example, suppose that your LDAP directory service will participate in a global directory structure and/or you are co-existing with a legacy X.500 directory service. Also suppose you are designing your directory tree for the fictional state of Verona, and that the following agencies and municipalities are immediately interested in participating in your state-wide LDAP directory service:

- The cities of New Port and Brighton
- The state agencies of the Legislature, Department of Natural Resources (DNR), and the Department of Motor Vehicles (DMV)
- The counties of Anoka and Brighton

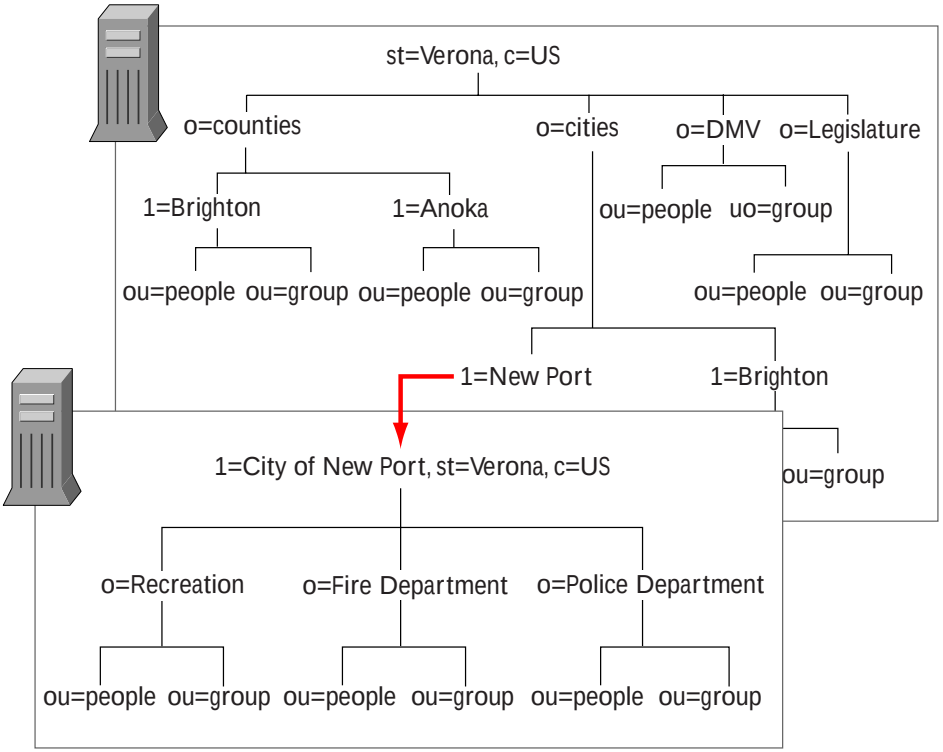
Also, further suppose that the city of New Port is already running their own LDAP service with a suffix of `l=City of New Port, st=Verona, c=US`.

Then do the following:

- Use a suffix for your directory service of `st=Verona, c=US`
- Create the following branch points:
  - `o=cities, st=Verona, c=US`
  - `o=counties, st=Verona, c=US`
  - `o=DNR, st=Verona, c=US`
  - `o=DMV, st=Verona, c=US`
  - `o=Legislature, st=Verona, c=US`
- Under the `ou=cities` branch, create the entries `l=New Port, o=cities, st=Verona, c=US` and `l=Brighton, o=cities, st=Verona, c=US`

- Under the `o=counties` branch, create the entries `l=Brighton, o=counties, st=Verona, c=US` and `l=Anoka, o=counties, st=Verona, c=US`.
- For each state agency, individual county, and individual city branch, treat the subtree as if it were a small organization. For an example of a small organization tree, see “A Small Organization” on page 128.
- For the `l=New Port, ou=cities, st=Verona, c=US` entry, create a smart referral to the server and port where the New Port directory service is running. Also, on the LDAP URL specify the suffix of `l=City of New Port, st=Verona, c=US`. For example:

  ```
  ldap://directory.newport.com/l=City of New Port,
  st=Verona, c=US
  ```



An International Corporation

Designing a directory tree for a global enterprise involves determining not only how to logically collect directory entries in descriptive subtrees, but also how to design the directory tree to support replication on a global scale. Exactly how you will approach this depends on a number of factors, including:

- whether you will support local data management
- the quality of the network connecting the various sites in your enterprise
- the nature of your directory data and your enterprise's need to replicate any piece of data everywhere
- whether all of the organizations in your enterprise can, or are willing to, share the same directory suffix (Some enterprises, such as corporations with subsidiaries, will find that a common suffix across the entire enterprise is difficult to create).

The following sections illustrate some international corporation directory designs.

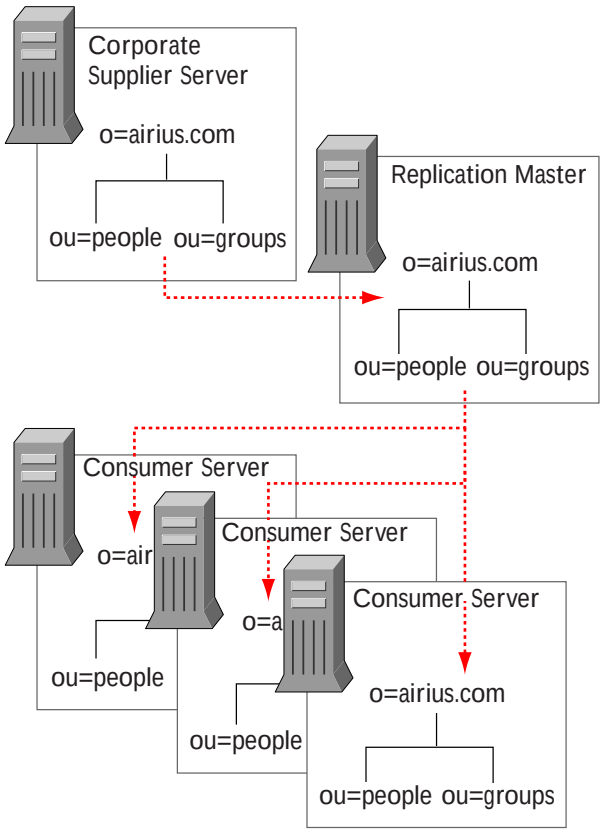
Single Suffix, Global Directory Tree Replication

Suppose you have an international organization, that you want to centrally manage all your data, and that you want all your data to be universally available. In this case, the directory tree that you would design would not be entirely different than what you would use for a small organization (see “A Small Organization” on page 128.)

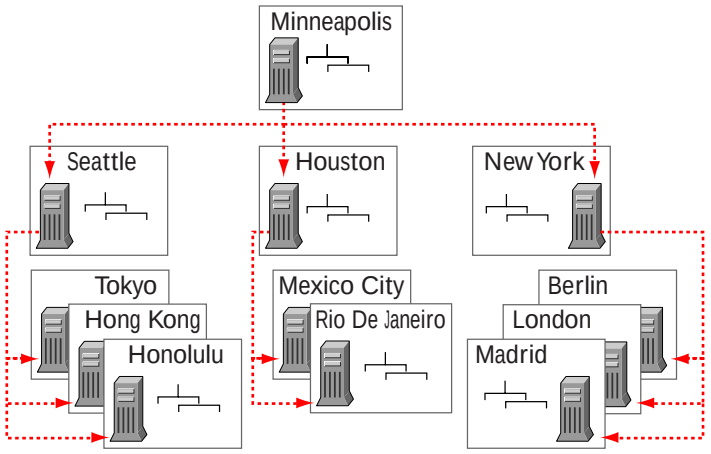
The only difficult part of this directory design is determining your replication strategy. This section provides an example replication strategy that will support a very large and complicated replication environment such as might be required by a multinational corporation (replication is discussed in detail in Chapter 7, “Planning Replication.”)

Because multinational corporations often encompass hundreds of thousands of employees in dozens of countries, you should consider using a replication master to support this environment. Essentially, a replication master is a

Directory Server whose only function is to replicate directory data to consumer servers. The replication master is the only consumer of your central supplier server. By using a replication master, you can free up your master Directory Server to handle all write activities for your enterprise.



Beyond the use of a replication master, you need to determine which locations in your enterprise have the best network connection to other locations in your enterprise. That is, use your replication master to replicate directory data to 2-6 locations for which the replication master has a fast, reliable connection. From those consumer servers, replicate directory data to other consumer servers with which they have a good network connection. Continue this strategy until you have complete coverage across your enterprise.

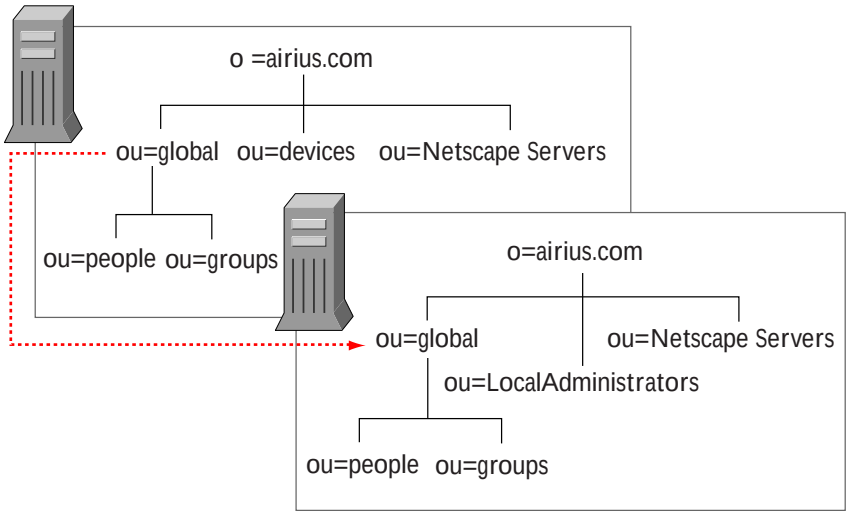


Single Suffix, Local Data Management

Extremely large multinational corporations are most likely to have to support local data management to some extent. This is because different countries will require different employment information, and different sites within your enterprise will require different kinds of information depending on the activities at that site.

The best way to support his kind of an enterprise is to determine what information is needed company-wide and what is needed only locally. If possible, segregate this information into subtrees that are needed locally and subtrees that are needed globally. To aid replication, group all of the trees needed globally in an `ou=global` tree and then replicate that single tree.

For example:



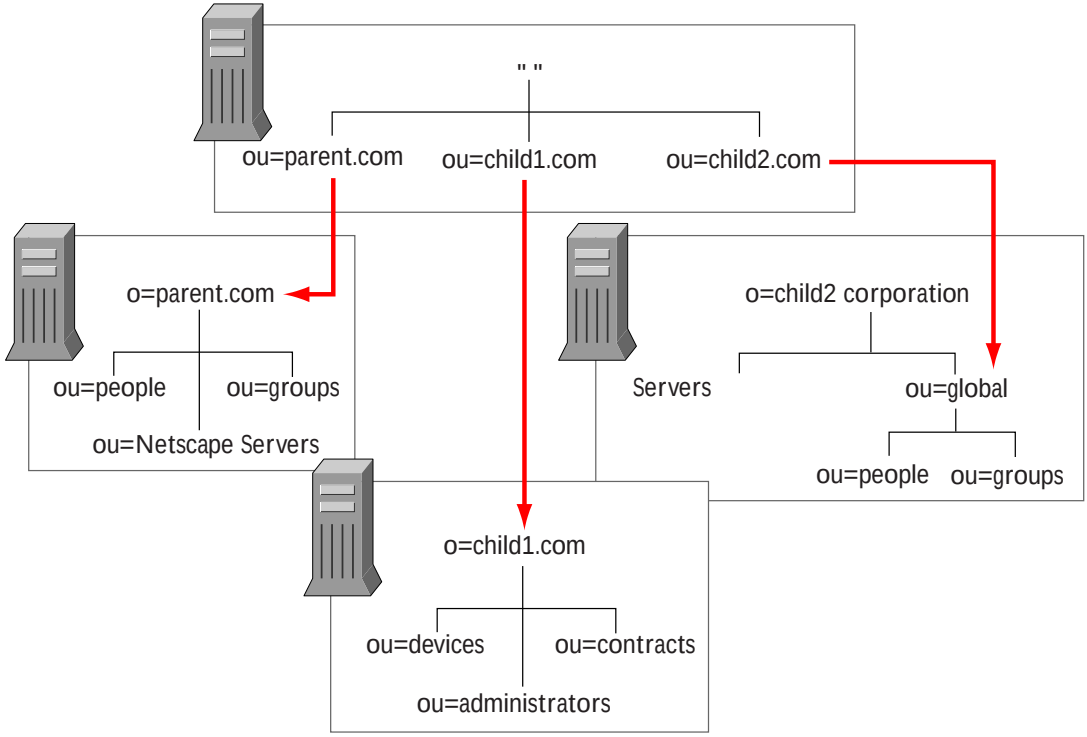
Multiple Suffixes, Local Data Management

Some very large organizations will not have the luxury of using the same suffix across the entire enterprise. As in the case of a state government (see “A State Government” on page 131), some organizations may have already deployed a directory service and in the process created a suffix that is different from the suffix used by your enterprise. In other situations, you may have a corporation with one name that owns multiple subsidiaries who are known by some other name.

A slightly different but related topic is the situation in which you allow each organization to manage their own unique directory service. In this case, each organization may be using the same suffix, or even the same directory tree design, but the nature of your enterprise/directory data may be such that you do not want to replicate directory data everywhere. Yet you may want your users to be able to (infrequently) locate data owned by other organizations.

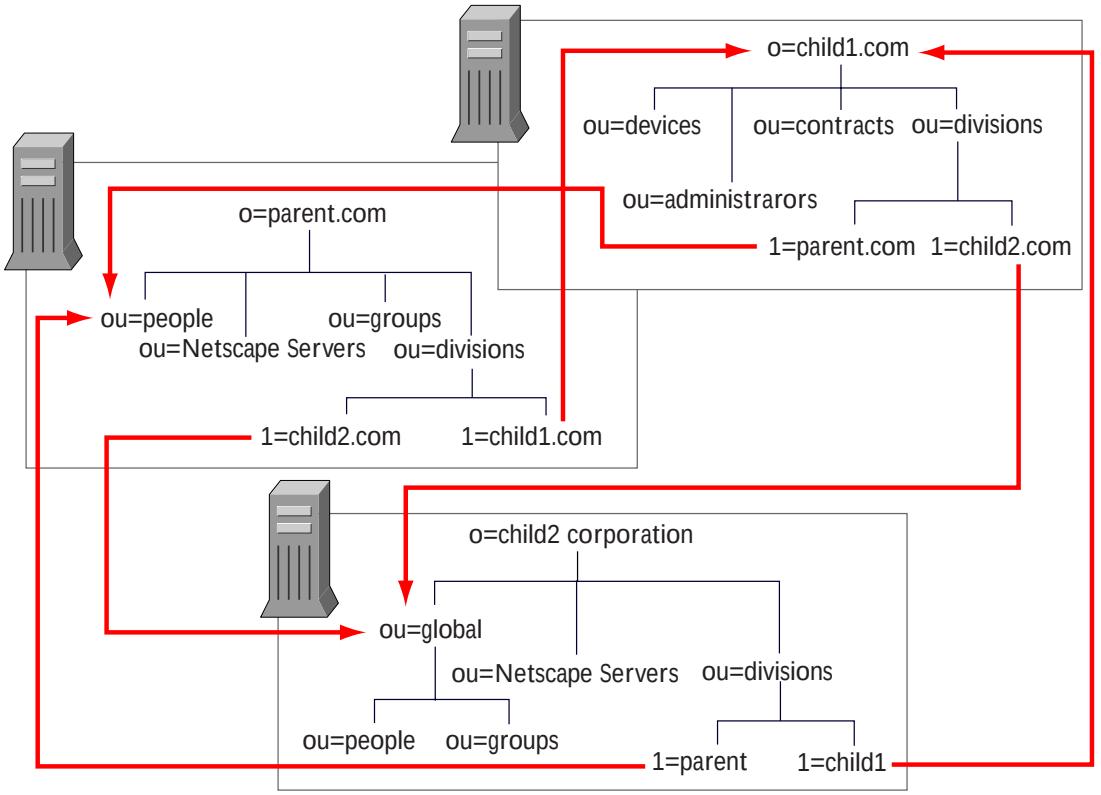
You can solve these kinds of problems by creating a directory of directories. A directory of directories is a directory tree that contains smart referrals to other Directory Servers around your enterprise.

For example, suppose you have a parent corporation (`parent.com`) with two subsidiaries (`child1.com` and `child2.com`). Suppose that everyone is running their own directory service and that everyone is using different suffixes. Then you can create a Directory Server that contains smart referrals to these different Directory Servers. People looking for entries in other organizations directory services can simply go to this central “directory of directories” and perform the search there. The smart referral mechanism will ensure that all search requests are routed and reformatted appropriately. In fact, the user performing the search may not ever know that the search request was referred to some other remote server.



Of course, a central server will cost you some additional overhead. You will have to find the hardware resources to contain the directory service, and then the directory has to be managed. However, in extremely large enterprises with a lot of different LDAP directory services, a centralized directory of directories may be the best approach.

For smaller environments, a better approach is to build the smart referrals directly into the local directory tree, as follows:



This kind of referral usage has the advantage of not forcing the user to go to a special directory service to perform cross-organization searches. The downside is that global directory tree searches will result in referrals being sent to the remote servers all the time. This will result in slower search performance and higher loads on both your network and servers. Depending on the number of

users of the various directory services, the number of Directory Servers, the quality of your network connections, and the robustness of your hardware, this kind of smart referral usage may not be appropriate.

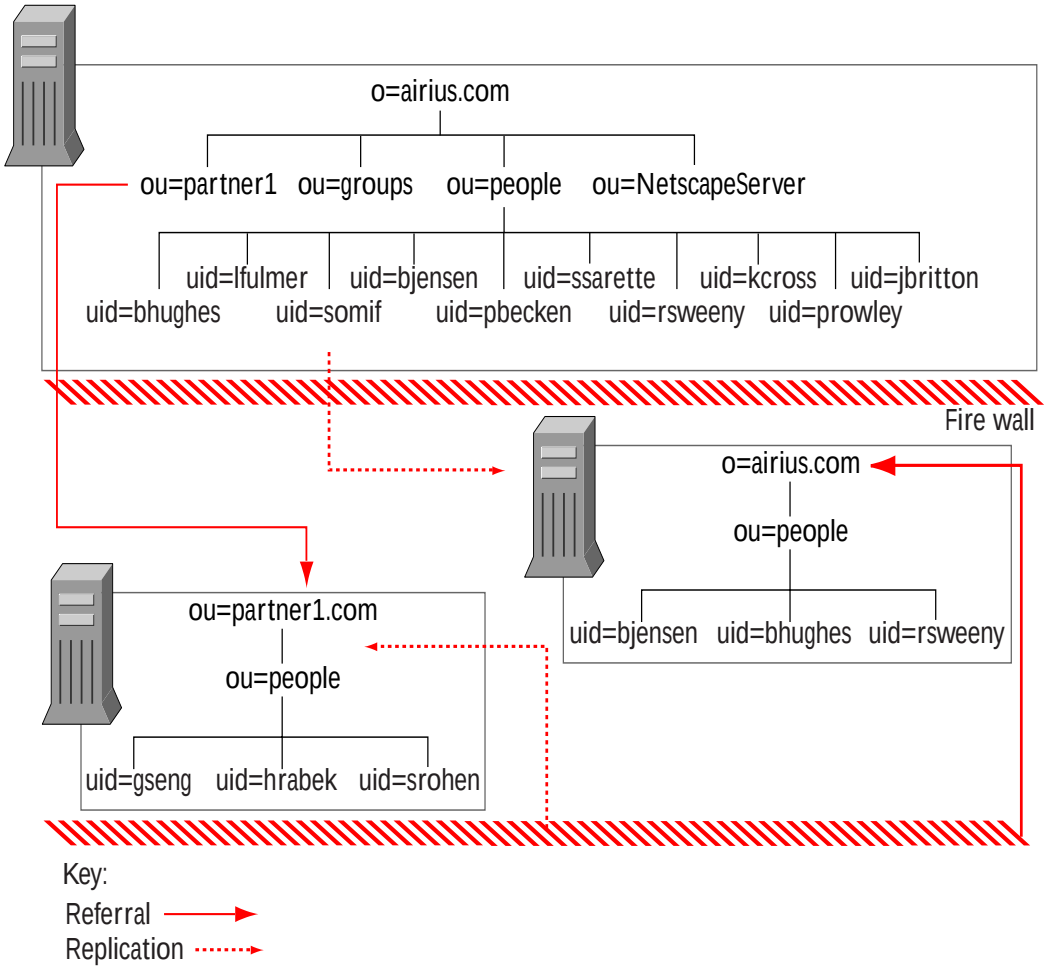
Enabling the Extranet

The *extranet* is a term that describes the extension of an enterprise's intranet to its trading partners. Basically, there may be a set of directory information that you might want your trading partners to access. Under these circumstances, there is also likely to be a set of information maintained by your trading partners that you want members of your enterprise to access. The question is: how to best go about doing this without violating security policies?

You should do the following:

1. Replicate only that set of information that you want to share with your trading partners to a Directory Server located outside your firewall. This information is likely to contain public contact information as well as private account or contract information of interest to your trading partners. You should consider this exterior Directory Server to be something of an independent directory service. As such, you should carefully plan the schema, data, and security policies that you will use for this unique service. Most likely you will find that the security policies and data management strategies that you have put into place for your internal directory service will not be appropriate for the needs of this external, more limited service.

2. On your internal directory service, create suffixes for each of your trading partner's own directory service. Each suffix should then have a corresponding root directory entry that is simply a smart referral to your trading partner's public directory service. This will allow members of your enterprise to search your internal directory service for information regarding your trading partners. All such searches will be referred to the appropriate Directory Server for handling.



To make this directory design work, you will have to coordinate authentication procedures with your trading partners. Every enterprise is likely to have a different approach to this problem, but the most elegant solution is probably to support certificate-based authentication for all extranet activities. For more information on using certificate-based authentication with your Directory Server, see the *Netscape Directory Server Administrator's Guide*.

Chapter

10

Extending Your Directory Service

It is possible that you will want to write custom programs to obtain all the functionality in your directory service that you need to support your environment. These programs can take the following forms:

- Custom filters to allow you to move data between the Netscape Directory Server and other directory services such as Novell NDS or X.500 directory services.
- Custom LDAP clients to help you manage data unique to your directory service.
- Custom server plug-ins to extend the capabilities of the Netscape Directory Server.

In this chapter you will find brief information on the circumstances under which you should consider creating custom programs to extend your directory service. You will also find brief information on the Application Programming Interfaces (APIs) that you can use to create Directory Server plug-ins and LDAP clients. This chapter includes the following sections:

- “Building Custom Tools for Data Migration” on page 142
- “Building Custom LDAP Clients” on page 143
- “The LDAP C and Java SDKs” on page 144
- “Writing Server Plug-Ins” on page 145

Building Custom Tools for Data Migration

As your directory service becomes more complex, it is likely that you will want to build custom clients and tools to move data in and out of your directory. Primarily you will do this if you are interoperating with a legacy directory or application that does not communicate over LDAP. Over time these kinds of interoperability problems should ease as more vendors start to support LDAP. However, for now you may either have to write a custom program to move data between your LDAP directory service and some legacy data store, or you will have to work with your vendor to get them to support LDAP in their application.

There are several basic approaches that you can take if you decide to write a custom program to move data in and out of the Directory Server:

1. You can work with LDIF to manipulate your directory contents into a form compatible with the legacy directory. This can mean writing applications to either change LDIF into a form that is compatible with your legacy directory, or taking output from your legacy directory and filtering it into LDIF format. This approach benefits from being the simplest to implement in that the standard Netscape LDAP command line tools work directly with LDIF for import and export operations. This means that you can use PERL scripts, batch files, or shell scripts to automate the import and export operations.

   However, this approach will result in relatively slow data migration. Importing and exporting directory data to and from LDIF is a slow operation compared to direct communication over LDAP. Also, scripts and interpreted languages generally run slower than compiled code. As a result, you should use this approach only if you can live with a batch approach to synchronization -- that is, only if you want to synchronize your directories once or twice a day.

   For information on LDIF and the Netscape LDAP command line tools, see the *Netscape Directory Server Administrator's Guide.*

2. Use the LDAP client APIs to write applications or applets that communicate with the Directory Server over LDAP. Netscape provides LDAP client SDKs in C and in Java that provide you access any of the standard Directory

Server features that are exposed over the LDAP v2 and LDAP v3 protocols (for example, you can write applications that search the directory or that modify data in the directory).

This approach requires a moderate amount of programming knowledge and possibly a fair amount of time. It also works best if the partner directory also supports an API set that allows you to move data in and out of it.

Building Custom LDAP Clients

While the Netscape Directory Server ships with several LDAP clients, you may find that these clients do not support your environment. There may be several reasons for this:

- The Netscape LDAP clients may not support your data management processes. One example of this is the condition known as referential integrity. Suppose you have some information, such as a phone number, stored on one object in your directory (such as a person entry). Suppose this same information is also stored in a second location (such as on an entry that represents a client contract). If this telephone number needs to change, you can manually change it everywhere it exists in the directory. Or you can write a custom client to search for and change the phone number everywhere that it exists in the directory.
- The standard Netscape clients are overly complicated for your needs. The HTTP to LDAP gateway in particular includes quite a bit of functionality that you may not want to expose to your general user audience. You may therefore decide to build some kind of a general-use LDAP client that supports just the subset of operations that you want your general directory population to use. For example, this client could support simple searches and some limited and well-defined set of directory modification operations.
- Your enterprise's schema requirements may be unique enough that it is easier to build a custom client to support it than to attempt to modify the standard Netscape LDAP clients to support your data needs.

As is the case for building custom data migration tools, there are two basic approaches that you can take when building custom LDAP clients:

1. Use the standard Netscape LDAP command line tools. This approach works best if you want to write some PERL-based CGIs to support a web interface to your directory. This method is used most frequently by sites who want a very simple interface to the directory.

2. Use the LDAP client APIs. While this approach requires some moderate programming skills, it offers you the ability to create efficient, flexible LDAP clients to support your unique requirements.

The LDAP C and Java SDKs

Netscape provides several Software Developer Kits (SDKs) to help you write custom clients or tools that speak directly to the Directory Server over LDAP. You can obtain a client SDK for either the C or the Java programming languages.

Both SDKs allow you to customize the way an LDAP operation is performed by using LDAP controls. For example, the Directory Server supports LDAP server controls that set up persistent search and that specify that the server is to sort the search results. As a result, you can write an LDAP client that sends a control to make the server sort the search requests.

Both SDKs also allow you to request additional operations from the server (beyond adding, modifying, deleting, and searching for entries). You can do this by defining extended operations -- a mechanism for defining additional services that you want to make available from your Directory Server. For example, you can write an LDAP client and a server plug-in that request and perform digitally signed operations.

For more information on the LDAP client C and Java SDKs, go to LDAP Directory Developer Central, which is located at the following URL:

`http://developer.netscape.com/one/directory/index.html`

Writing Server Plug-Ins

If you want to extend the capabilities of the Directory Server, you can write your own server plug-in. A *server plug-in* is a shared object or library (or a dynamic link library on Windows NT) containing your own functions.

When starting up, the Directory Server loads your server plug-in and calls your functions during the course of processing various LDAP requests.

You can write plug-in functions that:

- Validate data before the server performs an LDAP operation on the data. For example, you can write a plug-in function that checks new entries before they are added to the directory.
- Replace the existing database. For example, if you want to store directory data in a relational database, you can write plug-in functions that integrate your database with the Directory Server.
- Perform an action (that you define) after the server completes an LDAP operation. For example, you can write a plug-in function that sends email to a user after the user's entry has been modified.
- Implement extended operations available from your server.

For more information on the Directory Server plug-in API, go to the *Netscape Directory Server Programmer's Guide*, which is located at the following URL:

`http://developer.netscape.com/library/documentation/directory/plugin/index.htm`

Appendix

A

Quick Start

This appendix is intended for the administrator who wants to use the Directory Server to support basic Netscape server administration. You should read this appendix if the following describes you:

- You just want to get some Netscape server running, such as Enterprise, Messaging, or Collabra Server. That is, you are only using the Directory Server so that Netscape servers have a place to store user and group information.
- You are supporting employees that are close to one another within your network topology, and so network connections are not a major issue for you. That is, you are not trying to support employees across wide area networks, dial-up connections, or over unreliable network feeds between buildings.
- You are supporting a relatively small number of entries (fewer than 10,000).
- You do not want to develop a directory tree structure.

Straight out of the box the Directory Server supports basic Netscape Server activities. There are just a few things you need to consider before and during directory installation, and this appendix outlines those issues for you.

Note Make sure you have read Chapter 1, “Welcome to the Directory Server,” before continuing with this appendix. Chapter 1 contains information about basic Directory Server concepts that you need to understand before performing server installation.

A Word of Advice

By following the directions in this appendix, you can rapidly deploy a simple directory service for use with other Netscape servers. This is appropriate for testing or review purposes, or even for installing servers into a production environment if that environment is controlled and relatively secure.

However, for most production directory deployments, the environment is not so idealized as is assumed by this appendix. For this reason, if you are installing your server outside of the lab environment, you are strongly recommended to at least examine the data planning issues discussed in Chapter 3, “Planning Your Directory Data,” and the access-control issues raised in Chapter 5, “Planning Security Policies.”

In general, if you are deploying your directory service into a production environment, taking an afternoon to become familiar with the concepts and issues raised in the earlier portions this manual will go a long way towards helping you build a robust, secure, easily-administered directory service.

Planning Your Suffix Value

Before you get started, you must plan the suffix name under which you are going to store your directory entries. If your enterprise already has an Internet domain name registered for it, then use a suffix value of the following format:

```
o=<your domain name>
```

That is, if your enterprise’s Internet domain name is `Airius.com`, then use:

```
o=airius.com
```

as your suffix. If you do not have an Internet domain name registered for you enterprise, then simply use your company’s name.

For example:

```
o=airius corporation
```

Directory Tree Advice

Keep your directory tree as flat as possible. For a small or non-production directory service, there are no technical or administrative advantages to branching your directory tree. Consequently, simply create every entry at your directory's root level. That is, if your suffix is `o=airius.com`, then create all your entries using the following format:

```
uid=<some value>, o=airius.com
```

for people entries, and

```
cn=<some value>, o=airius.com
```

for group entries.

DN Advice

The one thing you absolutely must do is make sure every DN is unique. This can be one of the greatest challenges facing a directory administrator. This is because DNs traditionally begin with a common name (`cn` attribute). Common name-based distinguished names are not a problem for any kind of a directory entry other than entries that represent people. For people entries, common names are meant to be a person's full name. In theory, by using common name-based distinguished names you should be able to quickly know which person is represented by any given DN. Unfortunately, this strategy quickly fails because in any organization of even a moderate size, employees often have the same name.

To avoid naming collisions, use uid-based distinguished names for all person entries. Netscape servers require that every person entry managed by the servers are defined to have a unique user ID anyway. By using this user ID for your distinguished names, you can elegantly sidestep directory name collisions.

You should choose user IDs that are reasonably human readable; that is, do not use a random collection of letters and/or numbers for your user IDs. If your enterprise already has an email system, one possible solution is to select the left-most value on each person's email address for that person's user ID. That is, if a person has the email address:

```
bjensen@airius.com
```

then give that person's directory entry the following DN:

```
uid=bjensen, o=airius.com
```

Note It is recommended that you avoid using meaningless values such as an employee number for the `uid`. Using non-intuitive user IDs can create administrative burdens for you.

Also, remember that the information contained in a DN is essentially public information; anyone who can search for that entry can see all of the information contained in the DN. For this reason, limit your DNs to generic or public information.

For more information on designing directory trees and DN formats, see Chapter 6, "Directory Tree Design."

Configuration Directory Advice

Netscape 4.x servers now require a configuration directory in order to install. The configuration directory is a Directory Server that contains server configuration information. It is best defined as the Directory Server that contains the `o=Netscape Root` suffix and tree.

For extremely basic installations of the Directory Server, you can install the Directory Server such that it contains both the `o=Netscape Root` tree and your corporate directory data. However, for most installations, you should seperate the two so that they are two different server instances. Doing so will ease the server upgrade process in the future.

If you are making the configuration directory a seperate server instance from your primary production Directory Server, make sure you install the configuration directory on a port other than 389. Because the configuration

directory must be installed first, and because it is difficult to change the port that the configuration directory listens to after other Netscape servers have been installed, it is best to use a non-standard port for this Directory Server instance.

For more information on the configuration directory, see the *Netscape Directory Server Installation Guide.*

Creating Directory Entries

Once you have installed and configured your Directory Server, you will need to create directory entries. There are two basic ways you can do this.

- The first approach is useful if you want to rapidly create a directory with more than a few tens of entries. For mass imports of directory entries, create and import an LDIF file to your Directory Server. LDIF is the LDAP Data Interchange Format, and it allows you to create a flat ASCII file that represents the entries in your directory. This method of directory creation requires that you understand LDIF, the Directory Server schema (that is, the directory data model), and the schema used by the individual Netscape servers.

- A second, easier, but more time-consuming method of directory creation is to create directory entries one by one using a directory gateway such as the Users and Groups area in the Netscape Console. This method of directory creation is useful if you want to create only a few entries at a time.

The easiest approach to creating a production directory is to combine these two methods of directory creation as follows:

1. When you install your Directory Server, allow the installation process to create sample directory entries. This will cause the installation process to create a simple database with some simple directory branch points.

2. Use a directory gateway to create a few sample user and group entries. If you are using the messaging server with your directory, make sure to create some user accounts with mail information, as well as some mailing lists.

3. Export your directory to an LDIF file so that you can see the format of the individual sample entries.

4. Use this exported LDIF file as a template for creating your entire directory. Since you are likely to be creating large numbers of directory entries from some other source, such as an HR database, you will probably need to automate the creation of the LDIF file using a scripting language such as PERL.

5. Import your directory from your LDIF file.

For More Information

For information on how to install a Netscape Directory Server, see the *Netscape Directory Server Installation Guide.*

LDIF and Directory Server manager usage is described in the *Netscape Directory Server Administrator's Guide.*

Index

A

access
  anonymous 60, 73
  determining general types of 73
  precedence rule 62
  restricting by physical location 74
access control information (ACI) 61
  bind rules 66, 67, 68
  filtered rules 65
  format 66–71
  permission 66
  target 66, 67
  usage advice 69
  where to place 64
access control list (ACL) 61
  defined 61
  permissions 61
access rules
  overview 57
access-control
  branching to support 89
  planning 24
access-control information (ACI)
  filtered rules 89
  in the directory tree 89
  where to place 129
ACI, *See* access control information
ACL, *See* access control list
adding object classes 48
  strategies 48
allow permissions 63
  usage advice 63
analyzing the site survey 34
anonymous access 73
  for read 38
  overview 60
API, server 145
applications 31
architecture 16
attribute 42
  overview 44–46
  required and allowed 45
  values 45
attribute-data pair 29, 42
authentication 57, 59
  certification-based 58
  Directory Manager 61
  overview 57
  with Directory Server NT 59

B

base distinguished name 21
bind DN 57
bind rules 66, 67, 68
binding to the directory 57
  anonymously 60
  certificate-based 58
branch point 78
  DN attributes 84
    searching 86
    traditional 85
  for access-control 89
  for international trees 90
  for replication and referrals 88
  network names 88
  strategies 86
  usage advice 84

C

c attribute 90
C SDK 144
cascading replication 99
certificate-based authenticate 58
changelog 104
circular groups 73
clients 15
  API 142
  bind algorithm 58
  referrals and 125
  SDK 144
cn attribute 42, 43, 91, 149
commonName attribute 42, 43, 91, 93
configuration directory 150
consumer server 96, 97
consumer servers 97
consumer-initiated replication 102
  required directory entries 104
conventions, in this book 10
country attribute 64, 90
custom filters 141
  strategies 142
custom LDAP clients 141
  building 143
custom programs 141
  client SDKs 144
  clients, building 143
customizing the directory service 141
customizing the schema 40, 47–53
  being consistent 50
  FAQ 52

D

data access 37
data management
  local management example 135
  planning 24
  replication example 113
data mastering 34
  for multiple applications 35
  for replication 35, 109
data migration 142
data ownership 36
database 16
  access rules 57
  replacing 145
  with ISPs 81
database plug-in 16, 145
default permissions 62
deny permissions 62
  usage advice 63
  when to use 63
deployment advice 25
Directory Access Protocol (DAP) 14
directory applications 31
  browsers 31
  email 31
directory data 27–40
  access 37
  characteristics 29
  creating 151
  entry size 108
  examples of 30
  mastering 34
    for multiple applications 35
    for replication 35
  model 46
  ownership 36
  planning 28, 31
    site survey 33–40
  representation 42
  what not to include 30
directory deployment team 33
directory design
  activities 24
  advice 23
  examples
    extranet 139
    international corporation 133–139

multiple suffix, local data management 136
single suffix, global replication 133
single suffix, local data management 135
small organization 128
state government 131
directory entries
creating 151
directory information tree 17
Directory Manager 20
authentication 61
defined 61
directory of directories 137
directory schema 40
directory service 12–15
extending 141
global 14
LDAP 15
n+1 problem 13
Netscape solution 16
uses of 13
X.500 14, 131
directory suffix 78
country root point 80
planning 80
recommended 81, 148
directory tree 17, 77–93
branch point 78, 128, 131
DN attributes 84
searching 86
traditional 85
for access-control 89
for international trees 90
for replication and referrals 88
network names 88
strategies 86
usage advice 84
consumer 97
design advice 149
overview 78
planning 25
populating 151
replicated 98
suffix 78, 128, 131
country root point 80
planning 80
recommended 81, 148
supplier 97
distinguished name 18
name collision 92
avoiding 149
naming non-person entries 93
naming person entries 91
usage advice 149
DIT 17
DN, *See* distinguished name
DNS 13, 105
network sort 106
round robin 105

E

email applications 31
enterprise 12
examples
directory design 127–140
extranet 139
international corporation 133–139
multiple suffix, local data management 136
single suffix, global replication 133
single suffix, local data management 135
small organization 128
state government 131
replication
large sites 112
load balancing server traffic 114
local data management 113
messaging traffic 116
small sites 112
extended operations 144, 145
extending the directory service 141
extending the schema 47
FAQ 52
extranet

example 139
replication 100
smart referrals 123

F

filtered access control rules 65
fonts, in this book 10

G

global directory services 14
group attribute 64
groups
circular 73
examples 129
naming 93
nested 73
planning 25, 71
usage advice 73

H

highly available directory services 104

I

index 117
inetOrgPerson attribute 64
inheritance, in object classes 43
international enterprise
branching to support 90
interoperating with legacy directories 142

J

java SDK 144

L

LDAP, *See* Lightweight Directory Access Protocol
LDAP client API 142
LDAP Data Interchange Format (LDIF) 151
LDIF 151
legacy directory, interoperating with 142
Lightweight Directory Access Protocol (LDAP) 15
client 15
API 142
authentication 57
anonymous 60
certificate-based 58
custom 141
custom, building 143
custom operations 144
directory service architecture 15
directory services 15
extended operations 144
referral handling 125
server 15
load balancing
the network 108
the server 108
local data management 135

M

mail attribute 92
mastering directory data 34
for multiple applications 35
for replication 35
migrating directory data 142
multiple suffixes 79
with enterprises 82
with extranets 83
with ISPs 81

N

n+1 directory problem 13
name collision 92
avoiding 149
nested groups 73
Netscape Directory Server 11, 15–17

API 145
architecture 16
authentication 57
anonymous 60
certificate-based 58
capabilities 15
concepts 17–21
database 16
deployment advice 25
extended operations 144
extending 141, 145
load balancing 108
performance 107
plug-ins 141
security policy 56
Netscape Messaging Server
indexes, required 117
replication example 116
network names, branching to reflect 88
network sort 106
network, load balancing 108
non-person entries
naming 93

O

object class 42
adding new 48
inheritance 43
overview 43–44
standard 43
object class violation 45
organization attribute 64
organizationalPerson object class 43
organizationalUnit attribute 64
organizations, naming 93

P

passwords, NT Directory Server and 59
performance (server) 107
permissions 62
ACL and 61
allow 63
bind rules 66, 67, 68
default 62
deny 62
when to use 63
on ACIs 66
precedence rule 62
usage advice 63
persistent search 144
person entries, naming 91
planning
access-control 24
data management 24
directory contents 24
directory data 28
site survey 33–40
analyzing 34
documenting 39
directory tree 25
groups 25
referrals 25
replication 25
planning directory data 31
what to consider 32
plug-in 16, 141
server, writing 145
points of access 74
populating the directory 151
precedence rule 62

Q

quick deployment 147–152

R

RDN, *See* relative distinguished name 91
referrals 79, 119–126
branching to support 88
client handling 125
handling by LDAP client 125
overview 120

planning 25
smart referrals
client handling 125
how to use 123
overview 120
usages 123
when returned 120
relational database 145
relative distinguished name (RDN) 91
non-person entries 93
person entries 91
replication 95–104
agreement 102
architecture 96
branching to support 88
cascading 99
consumer server 96, 97
consumer-initiated 102
directory trees 98
examples
large sites 112
load balancing server traffic 114
local data management 113
messaging traffic 116
small sites 112
extranet 100
for high availability 104
initiating synchronization 102
load balancing 107
the network 108
the server 108
local availability 109
modifying data 97
multiple subtrees 100
overview 96
planning 25
single master 96
strategies 110
example 133
subtrees 100
supplier server 96, 97
supplier-initiated 102
replication master 133
root distinguished name 20
root DN 20
password 61
root DSE 78
root entry 19, 128, 131
root password 61

S

schema 40, 41–53
customizing 40, 47–53
being consistent 50
FAQ 52
deleting standard elements 47
extending 47
overview 42–46
schema checking 45
overview 47
SDK, *See* software developer kits
secure sockets layer 21, 58
security policy 38, 55
creating 71–75
overview 56
server database 16
server performance 107
server plug-in 145
site survey 33–40
analyzing 34
documenting 39
multinational enterprises 33
network capabilities 110
smart referral 79
client handling 125
example 132, 138
how to use 123
overview 120
usages 123
sn attribute 43
software developer kits (SDKs) 144
SSL (see Secure Sockets Layer)
standard object classes 43
streetAddress attribute 43

styles, in this book 10
subtree replication 100
  multiple subtrees 100
suffix 18, 78, 128, 131
  country root point 80
  multiple 79
    with extranets 83
    with ISPs 81
    with large enterprises 82
  planning 80
  recommended 81, 148
supplier DN 103
supplier servers 96, 97
  capabilities of 97
  synchronization and 102
supplier-initiated replication 102
  required directory entries 103
surname attribute 43

T

telephoneNumber attribute 43
terms, in this book 10
top object class 43

U

uid attribute 43, 92
user authentication 57
user IDs 150
userPassword attribute 43

X

X.500 14, 51, 84, 85
X.500, coexisting with 131