Deployment Guide

iPlanet Directory Server

Version 5.0

816-0800-01
April 2001

THIS DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

CETTE DOCUMENTATION EST FOURNIE “EN L'ÉTAT”, ET TOUTES CONDITIONS EXPRESSES OU IMPLICITES, TOUTES REPRÉSENTATIONS ET TOUTES GARANTIES, Y COMPRIS TOUTE GARANTIE IMPLICITE D'APTITUDE À LA VENTE, OU À UN BUT PARTICULIER OU DE NON CONTREFAÇON SONT EXCLUES, EXCEPTÉ DANS LA MESURE OÙ DE TELLES EXCLUSIONS SERAIENT CONTRAIRES À LA LOI.

Contents

About This Guide

Welcome to the iPlanet Directory Server, a product of the Sun-Netscape Alliance. Sun Microsystems, Inc. and American Online, Inc. formed the Sun-Netscape Alliance to provide easy-to-deploy, comprehensive enterprise and e-commerce solutions to business partners and other companies competing in the Net Economy.

This preface includes the following sections:

- Purpose of This Guide
- iPlanet Directory Server 5.0 Overview
- Conventions Used in This Guide
- Related Information

Purpose of This Guide

This guide provides you with a foundation for planning your directory. The information provided here is intended for directory decision makers, designers, and administrators.

The first chapter of this guide introduces basic directory concepts. Most of the remainder of the guide covers aspects of directory design, including schema design, the directory tree, topology, replication, and security. The last chapter provides sample deployment scenarios to help you plan simple deployments as well as complex deployments designed to support millions of users distributed worldwide.

iPlanet Directory Server 5.0 Overview

iPlanet Directory Server 5.0 provides the following key features:

- Multi-master replication—Provides a highly available directory service for both read and write operations. Multi-master replication can be combined with simple and cascading replication scenarios to provide a highly flexible and scalable replication environment.
- Chaining and referrals—Increases the power of your directory by storing a complete logical view of your directory on a single server while maintaining data on a large number of directory servers, transparently for clients.
- Roles and Class of Service—Provides a flexible mechanism for grouping and sharing attributes between entries in a dynamic fashion.
- Improved access control mechanism—Provides support for macros that dramatically reduce the number of access control statements used in the directory, and increase the speed of access control evaluation.
- Resource-limits by bind DN—Gives you the power to control the amount of server resources allocated to search operations based on the bind DN of the client.
- Multiple databases—Provides a simple way of breaking down your directory data to simplify the implementation of replication and chaining in your directory service.
- Password Policy and Account Lockout—Allows you to define a set of rules that govern how passwords and user accounts are managed in the directory server.
- SSL—Provides secure communications over the network including ciphers with up to 168-bit encryption.

The major components of iPlanet Directory Server 5.0 include:

- An LDAP server—The core of the directory service, provided by the `slapd` daemon, and compliant with the LDAP v3 Internet standards.
- Directory Server Console—An improved management console that dramatically reduces the effort of setting up and maintaining your directory service. The directory console is part of iPlanet Console, the common management framework for iPlanet servers.
- Directory Server Gateway—An HTTP to LDAP client that allows you to access directory data from a web browser.
- iPlanet Netscape Directory Express—A simple directory lookup tool that you can use right out of the box.
- SNMP Agent—Permits you to monitor your directory server in real time using the Simple Network Management Protocol (SNMP).

- Online backup and restore—Allows you to create backups and restore from backups while the server is running.

Conventions Used in This Guide

This guide uses the following conventions:

`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, Cautions and Tips mark important information. Make sure you read this information before continuing.

Throughout this guide you will see path references of the following form:

`/usr/iplanet/servers/slapd-`*serverID*`/...`

The `/usr/iplanet/servers` directory is the default installation directory. If you have installed the Directory Server in a different location, you should adapt the path accordingly. The *serverID* variable represents the server identifier you gave the server when you installed it. For example, if you gave the server an identifier of `phonebook`, then the actual path would be:

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

All paths specified in this manual are in UNIX format. If you are using a Windows NT version of Directory Server, use equivalent paths.

Related Information

The document set for iPlanet Directory Server also contains the following guides:

iPlanet Directory Server Installation Guide. Procedures for installing your Directory Server as well as procedures for migrating your Netscape Directory Server to iPlanet Directory Server.

iPlanet Directory Server Administrator's Guide. Procedures for the day-to-day maintenance of your directory service. Includes information on configuring server-side plug-ins.

iPlanet Directory Server Command-Line File Reference. Information about using the command-line scripts shipped with Directory Server.

iPlanet Schema Reference. Information about all the schema used in the iPlanet suite of products.

Other useful iPlanet information can be found at the following Internet locations:

- iPlanet release notes and other documentation:
  http://docs.iplanet.com/docs/manuals/
- iPlanet product status:
  http://www.iplanet.com/support/technical_resources/
- iPlanet Professional Services information:
  http://www.iplanet.com/services/pro_serv/index.html
- iPlanet developer information:
  http://developer.iplanet.com/
- iPlanet learning solutions:
  http://www.iplanet.com/learning/index.html
- iPlanet product data sheets:
  http://www.iplanet.com/products/index.html

Chapter 1

Introduction to Directory Server

iPlanet Directory Server provides a centralized directory service for your intranet, network, and extranet information. Directory Server integrates with existing systems and acts as a centralized repository for the consolidation of employee, customer, supplier, and partner information. You can extend Directory Server to manage user profiles and preferences, as well as extranet user authentication.

This chapter describes the basic ideas you need to understand before designing your directory. It includes the following sections:

- What is a Directory Service?
- Introduction to iPlanet Directory Server
- Directory Design Overview

What is a Directory Service?

The term directory service means the collection of software, hardware, and processes that store information about your enterprise, subscribers, or both and make that information available to users. A directory service consists of at least one instance of Directory Server and one or more directory client programs. Client programs can access names, phone numbers, addresses, and other data stored in the directory.

One common directory service is a Domain Name System (DNS) server. A DNS server maps a computer host name to an IP address. Thus, all of the computing resources (hosts) become clients of the DNS server. The mapping of host names allows users of your computing resources to easily locate computers on your network by remembering host names rather than numerical IP addresses.

However, the DNS server stores only two types of information: names and IP addresses. A true directory service stores virtually unlimited types of information.

iPlanet Directory Server stores all of your information in a single, network-accessible repository. The following are a few examples of the kinds of information you might store in a directory:

- Physical device information, such as data about the printers in your organization (where they reside, whether they are color or black and white, their manufacturer, date of purchase, and serial number)
- Public employee information, such as name, email address, and department
- Private employee information, such as salary, government identification numbers, home addresses, phone numbers, and pay grade
- Contract or account information, such as the name of a client, final delivery date, bidding information, contract numbers, and project dates

iPlanet Directory Server serves the needs of a wide variety of applications. It also provides a standard protocol and application programming interfaces (APIs) to access the information it contains.

The following sections describe global directory services and the Lightweight Data Access Protocol (LDAP).

About Global Directory Services

iPlanet Directory Server provides global directory services, meaning it provides information to a wide variety of applications. Until recently, many applications came bundled with their own proprietary databases. While a proprietary database can be convenient if you use only one application, multiple databases become an administrative burden if the databases manage the same information.

For example, suppose your network supports three different proprietary email systems, each system with its own proprietary directory service. If users change their passwords in one directory, the changes are not automatically replicated in the others. Managing multiple instances of the same information results in increased hardware and personnel costs, a problem referred to as the n + 1 directory problem.

A global directory service solves the n+1 directory problem by providing a single, centralized repository of directory information that any application can access. However, giving a wide variety of applications access to the directory requires a network-based means of communicating between the applications and the directory. iPlanet Directory Server uses LDAP (Lightweight Directory Access Protocol) to give applications access to its global directory service.

About LDAP

LDAP provides a common language that client applications and servers use to communicate with one another. LDAP is a “lightweight” version of the Directory Access Protocol (DAP) used by the ISO X.500 standard. DAP gives any application access to the directory via an extensible and robust information framework, but at an expensive administrative cost. DAP uses a communications layer that is not the Internet standard TCP/IP protocol and has complicated directory-naming conventions.

LDAP preserves the best features of DAP while reducing administrative costs. LDAP uses an open directory access protocol running over TCP/IP and uses simplified encoding methods. It retains the X.500 standard data model and can support millions of entries for a modest investment in hardware and network infrastructure.

Introduction to iPlanet Directory Server

iPlanet Directory Server includes the directory itself, the server-side software that implements the LDAP protocol, and a graphical user interface that allows end-users to search and change entries in the directory. Other LDAP clients are also available, including the directory managers in the iPlanet Console and the Address Book feature in Netscape Communicator 4.x. In addition, you can purchase other LDAP client programs or write your own using the LDAP client SDK included with the iPlanet Directory Server product.

Without adding other LDAP client programs, Directory Server can provide the foundation for your intranet or extranet. Every iPlanet server uses the directory as a central repository for shared server information, such as employee, customer, supplier, and partner data.

You can use Directory Server to manage extranet user-authentication, create access control, set up user preferences, and centralize user management. In hosted environments, partners, customers, and suppliers can manage their own portions of the directory, reducing administrative costs.

When you install Directory Server, the following components are installed on your machine:

- An LDAP server (Directory Server) with a plug-in interface

  The name of this process is `ns-slapd`.

- iPlanet Administration Server

For more information about the Administration Server, go to http://iplanet.com/products/iplanet_application/.

- iPlanet Console to manage the servers

  For more information about the iPlanet Console, see the Console documentation on http://docs.iplanet.com/docs/manuals/console.html.

- Directory Server Gateway (DSGW), a simple HTTP server for displaying user information

  The name of this process is `dsgw`. For more information about Directory Server Gateway, see the Gateway Customization Guide on http://docs.iplanet.com/docs/manuals/directory.html.

- Command-line tools for starting and stopping the server, importing and exporting data in the database, database reindexing, account inactivation and deactivation, LDIF merges, and kernel tuning

  For more information about the command-line tools, refer to the *iPlanet Directory Server Configuration, Command, and File Reference*.

- An SNMP monitor

  For more information about SNMP monitoring, refer to the *iPlanet Directory Server Administrator's Guide*.

This guide talks about the core Directory Server and the plug-ins it uses for doing its work. The next sections describe Directory Server in more detail. The topics discussed are:

- "Overview of Directory Server Architecture," on page 16
- "Directory Server Data Storage," on page 19

Overview of Directory Server Architecture

At installation, Directory Server contains the following:

- A server front-end responsible for network communications
- Plug-ins for server functions, such as access control and replication
- A basic directory tree containing server-related data.

The following sections describe each component of the directory in more detail.

Overview of the Server Front-End

The server front-end of Directory Server manages communications with directory client programs. Directory Server functions as a daemon on UNIX systems and as a service on Windows NT and Windows 2000. Multiple client programs can speak to the server in LDAP. They can communicate using LDAP over TCP/IP. The connection can also be protected with SSL/TLS, depending on whether the client negotiates the use of Transport Layer Security (TLS) for the connection.

When communication takes place with TLS, the communication is usually encrypted. In the future, when DNS security is present, TLS used in conjunction with secured DNS will provide confirmation to client applications that they are binding to the correct server. If clients have been issued certificates, TLS can be used by iPlanet Directory Server to confirm that the client has the right to access the server. TLS and its predecessor SSL are used throughout iPlanet Server products to perform other security activities such as message integrity checks, digital signatures, and mutual authentication between servers.

Multiple clients can bind to the server at the same time over the same network because the Directory Server is a multi-threaded application. 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 places around the network.

Server Plug-ins Overview

Directory Server relies on plug-ins. A plug-in is a way to add functionality to the core server. For example, a database is a plug-in.

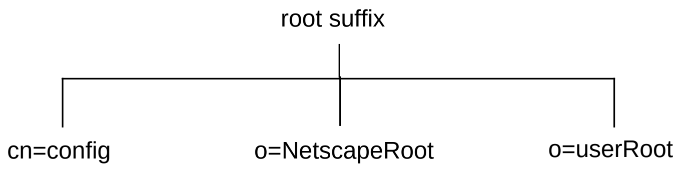
A plug-in can be disabled. When disabled, the plug-in’s configuration information remains in the directory but its function will not be used by the server. Depending upon the what you want your directory to do, you can choose to enable any of the plug-ins provided with Directory Server.

iPlanet Professional Services can write custom plug-ins for your Directory Server deployment. Contact iPlanet Professional Services for more information.

Overview of the Basic Directory Tree

The directory tree, also known as a directory information tree or DIT, mirrors the tree model used by most file systems, with the tree’s root, or first entry, appearing at the top of the hierarchy. At installation, Directory Server creates a default directory tree.

The default directory tree appears as follows:



The root of the tree is called the root suffix. For information about naming the root suffix, refer to “Choosing a Suffix,” on page 58.

At installation, the directory contains up to four subtrees under your root suffix:

- `cn=config`

  This subtree contains information about the server’s internal configuration.

- `o=NetscapeRoot`

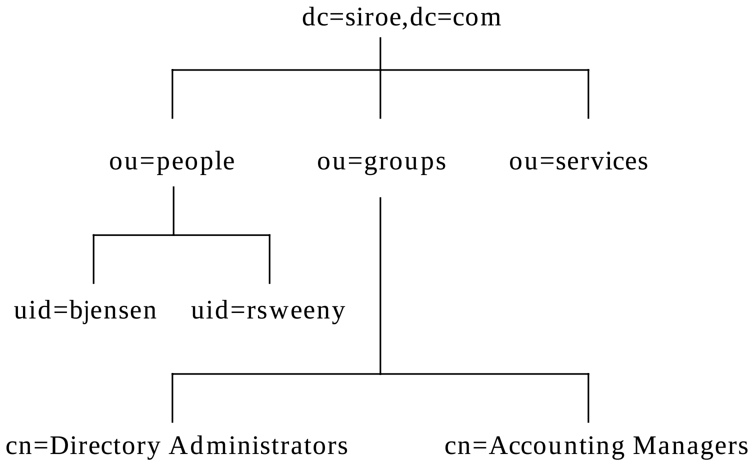
  This subtree contains the configuration information of other iPlanet servers, such as iPlanet Administration Server. The Administration Server takes care of authentication and all actions that cannot be performed through LDAP (such as starting or stopping).

- `o=userRoot`

  During installation, a user database is created by default. Its default name is `o=userRoot`. You can choose to populate it at installation time, or populate later.

NOTE When you install another instance of Directory Server, you can specify that it does not contain the `o=NetscapeRoot` information, that it uses the *configuration directory* (or the `o=NetscapeRoot` subtree) present on another server. See the *iPlanet Directory Server Installation Guide* for more information about deciding upon the location of your configuration directory.

You can build on the default directory tree to add any data relevant to your directory installation. An example of a directory tree for Siroe Corporation follows:



For more information about directory trees, refer to Chapter 4, “Designing the Directory Tree.”

Directory Server Data Storage

Your directory data is stored in an LDBM database. The LDBM database is implemented as a plug-in that is automatically installed with the directory and is enabled by default.

The database is the basic unit of storage, performance, replication, and indexing. You can do operations like importing, exporting, backing up, restoring, and indexing on the database.

By default, Directory Server uses a single database to contain the directory tree. This database can manage millions of entries. The default database supports advanced methods of backing up and restoring your data, so that your data is not at risk.

You can choose to use multiple databases to support your Directory Server. You can distribute your data across the databases, allowing the server to hold more data than can be stored in a single database.

The following sections describe how a directory database stores data.

About Directory Entries

LDIF is a standard text-based format for describing directory entries. An entry is a group of lines in the LDIF file that contains information about an object, such as a person in your organization or a printer on your network. Information about the entry is represented in the LDIF file by a set of attributes and their values. Each entry has an object class attribute that specifies the kind of object the entry describes and defines the set of additional attributes it contains. Each attribute describes a particular trait of an entry.

For example, an entry might be of an object class `organizationalPerson`, indicating that the entry represents a person within a particular organization. This object class allows the `givenname` and `telephoneNumber` attributes. The values assigned to these attributes give the name and phone number of the person represented by the entry.

iPlanet Directory Server also uses read-only attributes that are calculated by the server. These attributes are called operational attributes. There are also some operational attributes that can be set by the administrator, for access control and other server functions.

Entries are stored in a hierarchical structure in the directory tree. In LDAP, you can query an entry and request all entries below it in the directory tree. This subtree is called the base distinguished name, or base DN. For example, if you make an LDAP search request specifying a base DN of `ou=people, dc=siroe,dc=com`, then the search operation examines only the `ou=people` subtree in the `dc=siroe,dc=com` directory tree.

However, all entries are not automatically returned in response to an LDAP search. This is because Directory Server 5.0 introduces a new kind of entry, entries of the object class `ldapsubentry`. An `ldapsubentry` entry represents an administrative object, for example entries used to define a role or a class of service are of the `ldapsubentry` type. Entries of type `ldapsubentry` are not returned in response to normal search requests. To receive these entries, clients need to search specifically for entries of the `ldapsubentry` object class.

For more information about roles, see “About Roles,” on page 71. For more information about class of service, see “About Class of Service,” on page 72.

Distributing Directory Data

When you store various parts of your tree in separate databases, your directory can process client requests in parallel, improving performance. You can also store your databases on different machines, to further improve performance.

To connect your distributed data, you can create a special entry in a subtree of your directory. All LDAP operations attempted below this entry are sent to a remote machine where the entry is actually stored. This method is called chaining.

Chaining is implemented in the server as a plug-in. The plug-in is enabled by default. Using this plug-in, you create database links, special entries that point to data stored remotely. When a client application requests data from a database link, the database link retrieves the data from the remote database and returns it to the client.

Directory Design Overview

The previous sections described directory services in general and the iPlanet Directory Server in particular. Now it is time to consider the design of your own directory service.

Planning your directory service before actual deployment is the most important task to ensure the success of your directory. During your directory design you will gather data about your directory requirements, such as environment and data sources, your users, and the applications that will use your directory. With this data, you can design a directory service that meets your needs.

However, keep in mind that the flexibility of iPlanet Directory Server allows you to rework your design to meet unexpected or changing requirements, even after you deploy Directory Server.

Design Process Outline

The remainder of this guide divides the design process into six steps:

- How to Plan Your Directory Data.

  Your directory will contain data, such as user names, telephone numbers, and group details. Chapter 2, “How to Plan Your Directory Data,” helps you analyze the various sources of data in your organization and understand their relationship with one another. It describes the types of data you might store in your directory, and other tasks you need to perform to design the contents of your Directory Server.

- How to Design the Schema.

  Your directory is designed to support one or more directory-enabled applications. These applications have requirements of the data you store in your directory, such as format. Your directory schema determines the characteristics of the data stored in your directory. Chapter 3, “How to Design the Schema,” introduces the standard schema shipped with iPlanet Directory Server, describes how to customize the schema, and provides tips for maintaining consistent schema.

- Designing the Directory Tree.

  Once you decide what data your directory contains, you need to organize and reference that data. This is the purpose of the directory tree. In Chapter 4, “Designing the Directory Tree,” the directory tree is introduced and you are guided through the design of your data hierarchy. Sample directory tree designs are also provided.

- Designing the Directory Topology.

  Topology design involves determining how you divide your directory tree among multiple physical Directory Servers and how these servers communicate with one another. Chapter 5, “Designing the Directory Topology,” describes the general principles behind topology design, discusses using multiple databases, describes the mechanisms available for linking your distributed data together, and explains how the directory itself keeps track of distributed data.

- Designing the Replication Process.

  With replication, multiple Directory Servers maintain the same directory data to increase performance and provide fault tolerance. Chapter 6, “Designing the Replication Process,” describes how replication works, what kinds of data you can replicate, common replication scenarios, and tips for building a highly available directory service.

- Designing a Secure Directory.

  Finally, you need to plan how to protect the data in the directory and design the other aspects of your service to meet the security requirements of your users and applications. Chapter 7, “Designing a Secure Directory,” describes common security threats, provides an overview of security methods, discusses the steps in analyzing your security needs, and provides tips for designing access controls and protecting the integrity of your directory data.

Deploying Your Directory

After you have designed your directory service, you start the deployment phase. The deployment phase consists of the following steps:

- Piloting Your Directory
- Putting Your Directory Into Production

Piloting Your Directory

The first step of the deployment phase is installing a server instance as a pilot and testing whether your service can handle your user load. If the service is not adequate as it is, adjust your design and pilot it again. Adjust your pilot design until you have a robust service you can confidently introduce to your enterprise.

For a comprehensive overview of creating and implementing a directory pilot, refer to *Understanding and Deploying LDAP Directory Services* (T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999).

Putting Your Directory Into Production

Once you have piloted and tuned the service, you need to develop and execute a plan for taking the directory service from a pilot to production. Create a production plan that includes the following:

- An estimate of the resources you need
- A list of the tasks you must perform before installing servers
- A schedule of what needs to be accomplished and when
- A set of criteria for measuring the success of your deployment

For information on installing your directory service, refer to the *iPlanet Directory Server Installation Guide*. For information on administering and maintaining your directory, refer to the *iPlanet Directory Server Administrator's Guide*.

Other General Directory Resources

For more information about directories, LDAP, and LDIF, take a look at the following:

- RFC 2849: The LDAP Data Interchange Format (LDIF) Technical Specification http://www.ietf.org/rfc/rfc2849.txt

- RFC 2251: Lightweight Directory Access Protocol (v3)
  http://www.ietf.org/rfc/rfc2251.txt
- *Understanding and Deploying LDAP Directory Services.*
  T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999.

Chapter 2

How to Plan Your Directory Data

The data stored in your directory may include user names, email addresses, telephone numbers, and information about groups users are in, or it may contain other types of information. The type of data in your directory determines how you structure the directory, to whom you allow access to the data, and how this access is requested and granted.

This chapter describes the issues and strategies behind planning your directory's data. It includes the following sections:

- Introduction to Directory Data
- Defining Your Directory Needs
- Performing a Site Survey

Introduction to Directory Data

Some types of data are better suited to your directory than others. Ideal data for a directory has some of the following characteristics:

- It is read more often than written.

  Because directory are tuned for read operations, write operations slow your server's performance down.

- It is expressible in attribute-data format (for example, `surname=jensen`).
- It is 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 will be accessed from more than one physical location.

For example, an employee's preference settings for a software application may not seem to be appropriate for the directory because only a single instance of the application needs access to the information. However, if the application is capable of reading preferences from the directory and users might want to interact with the application according to their preferences from different sites, then it is very useful to include the preference information in the directory.

What Your Directory Might Include

Examples of data you can put in your directory are:

- Contact information, such as telephone numbers, physical addresses, and email addresses.
- Descriptive information, such as an employee number, job title, manager or administrator identification, and job-related interests.
- Organization 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, 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, and pricing information.
- Individual software preferences or software configuration information.
- Resource sites, such as pointers to web servers or the file system of a certain file or application.

If you are going to use Directory Server for more than just server administration, then you have to decide what other types of information you want to store in your directory. For example, you might include some of the following types of information:

- Contract or client account details
- Payroll data
- Physical device information
- Home contact information
- Office contact information for the various sites within your enterprise

What Your Directory Should Not Include

Directory Server is excellent for managing large quantities of data that client applications read and occasionally write, but it is not designed to handle large, unstructured objects, such as images or other media. These objects should be maintained in a file system. However, your directory can store pointers to these kinds of applications through the use of FTP, HTTP, or other types of URL.

Because the directory works best for read operations, you should avoid placing rapidly changing information in the directory. Reducing the number of write operations occurring in your directory improves overall search performance.

Defining Your Directory Needs

When you design your directory data, think not only of the data you currently require but also what you may include in your directory in the future. Considering the future needs of your directory during the design process influences how you you structure and distribute the data in your directory.

As you plan, consider these points:

- What do you want to put in your directory today? What immediate problem do you hope to solve by deploying a directory? What are the immediate needs of the directory-enabled application you use?
- What do you want to put in your directory in the near future? For example, your enterprise might use an accounting package that does not currently support LDAP, but that you know will be LDAP-enabled in the near future. You should identify the data used 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 store in your directory in the future? For example, if you are a hosting environment, perhaps future customers will have different data requirements from your current customers. Maybe future customers will want to use your directory to store JPEG images. 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 sources you might otherwise not have considered.

Performing a Site Survey

A site survey is a formal method for discovering and characterizing the contents of your directory. Budget plenty of time for performing a site survey, as data is the key to your directory architecture.The site survey consists of the following tasks, which are described briefly here and in more detail next:

- Identify the applications that use your directory.

  Determine the directory-enabled applications you deploy and their data needs.

- Identify data sources.

  Survey your enterprise and identify sources of data (such as NT or Netware directories, PBX systems, Human Resources databases, email systems, and so forth).

- Characterize the data your directory needs to contain.

  Determine what objects should be present in your directory (for example people or groups), and what attributes of these objects you need to maintain in your directory (such as user name and passwords).

- Determine the level of service you need to provide.

  Decide how available your directory data needs to be to client applications and design your architecture accordingly. How available your directory needs to be affects how you replicate data and configure chaining policies to connect data stored on remote servers.

  For more information about replication, refer to Chapter 6, “Designing the Replication Process” on page 97. For more information on chaining, refer to Chapter 5, “Designing the Directory Topology” on page 78.

- Identify a data master.

  A data master contains the primary source for directory data. This data might be mirrored to other servers for load balancing and recovery purposes. For each piece of data, determine its data master.

- Determine data ownership.

  For each piece of data, determine the person responsible for ensuring that the data is up-to-date.

- Determine data access.

If you import data from other sources, develop a strategy for both bulk imports and incremental updates. As a part of this strategy, try to master data in a single place, and limit the number of applications that can change the data. Also, limit the number of people who write to any given piece of data. A smaller group ensures data integrity while reducing your administrative overhead.

- Document your site survey.

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. This team performs the site survey.

Corporations generally have a human resources 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 perform the survey. Furthermore, directly involving all the affected organizations can help build acceptance for the migration from local data stores to a centralized directory.

Identifying the Applications that Use Your Directory

Generally, the applications that access your directory and the data needs of these applications drive the planning of your directory contents. Some of the common applications that use your directory include:

- Directory browser applications, such as online telephone books. Decide what information (such as email addresses, telephone numbers, and employee name) your users need and make sure you include it in the directory.
- Email applications, especially email servers. All email servers require email addresses, user names, and some routing information to be available in the directory. Others, however, require more advanced information such as the place on disk where a user’s mailbox is stored, vacation notification information, and protocol information (IMAP versus POP, for example).
- Directory-enabled human resources applications. These require more personal information such as government identification numbers, home addresses, home telephone numbers, birth dates, salary, and job title.

When you examine the applications that will use your directory, look at the types of information each application uses. The following table gives an example of applications and the information used by each:

Table 2-1 Application Data Needs

Application	Class of Data	Data
Phone book	People	Name, email address, phone number, user ID, password, department number, manager, mail stop
Web server	People, groups	User ID, password, group name, groups members, group owner
Calendar server	People, meeting rooms	Name, user ID, cube number, conference room name

Once you identify the applications and information used by each application, you can see that some types of data are used by more than one application. Doing this kind of exercise during the data planning stage can help you avoid data redundancy problems in your directory and see more clearly what data your directory dependant applications require.

The final decision you make about the types of data you maintain in your directory and when you start maintaining it is affected by these factors:

- The data required by your various legacy applications and your user population.
- The ability of your legacy applications to communicate with an LDAP directory.

Identifying Data Sources

To identify all of the data that you want to include in your directory, you should perform a survey of your existing data stores. Your survey should include the following:

- Identify organizations that provide information.

  Locate all the organizations that manage information essential to your enterprise. Typically this includes your information services, human resources, payroll, and accounting departments.

- Identify the tools and processes that are information sources.

Some common sources for information are networking operating systems (Windows NT, Novell Netware, UNIX NIS), email systems, security systems, PBX (telephone switching) systems, and human resources applications.

- Determine how centralizing each piece of data affects the management of data.

  You may find that centralized data management requires new tools and new processes. Sometimes centralization requires increasing staff in some organizations while decreasing staff in others.

During your survey, you may come up with a matrix that resembles the following table, identifying all of the information sources in your enterprise:

Table 2-2 Information Sources

Data Source	Class of Data	Data
Human resources database	People	Name, address, phone number, department number, manager
Email system	People, Groups	Name, email address, user ID, password, email preferences
Facilities system	Facilities	Building names, floor names, cube numbers, access codes

Characterizing Your Directory Data

All of the data you identify for inclusion in your directory can be characterized according to the following general points:

- Format
- Size
- Number of occurrences in various applications
- Data owner
- Relationship to other directory data

You should study each piece of data you plan to include in your directory to determine what characteristics it shares with the other pieces of data. This helps save time during the schema design stage, described in more detail in Chapter 3, “How to Design the Schema.”

For example, you can create a table that characterizes your directory data as follows:

Table 2-3 Directory Data Characteristics

Data	Format	Size	Owner	Related to
Employee Name	Text string	128 characters	Human resources	User's entry
Fax number	Phone number	14 digits	Facilities	User's entry
Email address	Text	Many character	IS department	User's entry

Determining Level of Service

The level of service you provide depends upon the expectations of the people who rely on directory-enabled applications. To determine the level of service each application expects, first determine how and when the application is used.

As your directory evolves, it may need to support a wide variety of service levels, from production to mission critical. It can be difficult raising the level of service after your directory is deployed, so make sure your initial design can meet your future needs.

For example, if you determine that you need to eliminate the risk of total failure, you might consider using a multi-master configuration, in which several masters exist for the same data. The next section discusses determining data masters in more detail.

Considering a Data Master

The data master is the server that is the master source of data. Consider which server will be the data master when your data resides in more than one physical site. For example, when you use replication or use applications that cannot communicate over LDAP, data may be spread over more than one site. If piece of data is present in more than one location, you need to decide which server has the master copy and which server receives updates from this master copy.

Data Mastering for Replication

iPlanet Directory Server allows you to contain master sources of information on more than one server. If you use replication, decide which server is the master source of a piece of data. iPlanet Directory Server 5.0 supports multi-master configurations, in which more than one server are masters sources for the same pice of data. For more information about replication and multi-master replication, see "Designing the Replication Process," on page 97.

In the simplest case, put a master source of all of your data on two Directory Servers and then replicate that data to one or more consumer servers. Having two master servers provides safe failover in the event that a server goes off-line. In more complex cases, you may want to store the data in multiple databases, so that the entries are mastered by a server close to the applications which will update or search that data.

Data Mastering Across Multiple Applications

You also need to consider the master source of your data if you have applications that communicate indirectly with the directory. Keep the processes for changing data, and the places from which you can change data, as simple as possible. Once you decide on a single site to master a piece of data, use the same site to master all of the other data contained there. A single site simplifies troubleshooting if your databases get out of sync across your enterprise.

Here are some ways you can implement data mastering:

- Master the data in both the directory and all applications that do not use the directory.

  Maintaining multiple masters does not require custom scripts for moving data in and out of the directory and the other applications. However, if data changes in one place, someone has to change it on all the other sites. Maintaining master data in the directory and all applications not using the directory can result in data being unsynchronized across your enterprise (which is what your directory is supposed to prevent).

- Master the data in the directory and synchronize data with other applications using iPlanet MetaDirectory.

  Maintaining a data master that synchronizes with other applications makes the most sense if you are using a variety of different directory and database applications. Contact your iPlanet sale representative for more information about iPlanet MetaDirectory, or go to the iPlanet website at http://www.iplanet.com/.

- Master the data in some application other than the directory and then write scripts, programs, or gateways to import that data into the directory.

  Mastering data in non-directory applications 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 only for lookups (for example, for online corporate telephone books).

How you maintain master copies of your data depends on your specific needs. However, regardless of the how you maintain data masters, keep it simple and consistent. For example, you should not attempt to master data in multiple sites, then automatically exchange data between competing applications. Doing so leads to a “last change wins” scenario and increases your administrative overhead.

For example, suppose you want to manage an employee’s home telephone number. Both the LDAP directory and a human resources database store this information.The human resources application is LDAP enabled, so you can write an automatic application that transfers data from the LDAP directory to the human resources database, and vice versa. However, if you attempt to master changes to that employee’s telephone number in both the LDAP directory and the human resources data, then the last place where the telephone number was changed overwrites the information in the other database. This is acceptable as long as the last application to write the data had the correct information. But if that information was old or out of date (perhaps because, for example, the human resources data was reloaded from a backup), then the correct telephone number in the LDAP directory will be deleted.

Determining Data Ownership

Data ownership refers to the person or organization responsible for making sure the data is up-to-date. During the data design, decide who can write data to the directory. Some common strategies for deciding data ownership follow:

- 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 subset of information might include their 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 approach makes your organization's administrators your directory content managers.
- Create roles that give groups of people read or write access privileges.

  For example, you might create roles for human resources, finance, or accounting. Allow each of these roles to have read access, write access, or both to the data needed by the group, such as salary information, government identification number (in the US, social security number), and home phone numbers and address.

  For more information about roles and grouping entries, refer to "Grouping Directory Entries," on page 70.

As you 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 an information systems (IS) or directory management group to have write access to employee passwords. You may also want the employees themselves to have write access to their own passwords. While you generally must give multiple people write access to the same information, try to keep this group small and easy to identify. Keeping the group small helps ensure your data's integrity.

The iPlanet Delegated Administrator can be used to provide partitioned account management and delegate administration responsibility for users to individuals in different roles across the organization. For more information, contact your iPlanet sales representative or go to the iPlanet Web site at http://www.iplanet.com.

For information on setting access control for your directory, see Chapter 7, "Designing a Secure Directory," on page 121.

Determining Data Access

After determining data ownership, decide who can read each piece of data. For example, you may decide to store an employee's home phone number in your directory. This data may be useful for a number of organizations, including the employee's manager and human resources. You may want the employee to be able to read this information for verification purposes. However, home contact information can be considered sensitive. Therefore, you must determine if you want this kind of data to be widely available across your enterprise.

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

- Can the data be read anonymously?

  The LDAP protocol supports anonymous access, and allows easy lookups for common information such as office sites, email addresses, and business telephone numbers. However, anonymous access gives anyone with access to the directory access to the common information. Consequently, you should use anonymous access sparingly.

- Can the data be read widely across your enterprise?

  You can set up access control so that the client must log in to (or bind to) the directory to read specific information. Unlike anonymous access, this form of access control ensures that only members of your organization can view directory information. It also allows you to capture login information in the directory's access log, so you have a record of who accessed the information.

  For more information about access controls, refer to "Designing Access Control," on page 136.

- Can you identify a group of people or applications that need to read the data?

  Anyone who has write privileges to the data generally also needs read access (with the exception of write access to passwords). You may also have data specific to a particular organization or project group. Identifying these access needs helps you determine what groups, roles, and access controls your directory needs.

  For information about groups and roles, see Chapter 4, "Designing the Directory Tree" on page 57. For information about access controls, see Chapter 7, "Designing a Secure Directory," on page 121.

As you make these decisions for each piece of directory data, you define a security policy for your directory. Your decisions depend upon 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, you may feel freer to support anonymous access than if you are placing your directory directly on the Internet.

In many countries, data protection laws govern how enterprises must maintain personal information, and restrict who has access to the personal information. For example, the laws may prohibit anonymous access to addresses and phone numbers, or may require that users have the ability to view and correct information in entries which represent them. Be sure to check with your organization's legal department to ensure that your directory deployment follows all necessary laws for the countries in which your enterprise operates.

The creation of a security policy and the way you implement it is described in detail in Chapter 7, “Designing a Secure Directory,” on page 121.

Documenting Your Site Survey

Because of the complexity of data design, document the results of your site surveys. During each step of the site survey we have suggested simple tables for keeping track of your data. Consider building a master table that outlines your decisions and outstanding concerns. You can build this table with the word-processing package of your choice, or use a spreadsheet so that the table’s contents can easily be sorted and searched.

A simple example of a table follows. The table identifies data ownership and data access for each piece of data identified by the site survey.

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 log in)	No	Yes
Office phone number	Facilities	Phone switch	Read-only	Yes (anonymous)	No	No

Looking at the row representing the employee name data, we see the following:

- Owner

  Human Resources owns this information and therefore is responsible for updating and changing it.

- Master Server / Application

  The PeopleSoft application manages employee name information.

- 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 with access to the directory.

- HR Writable

  Members of the human resources group can change, add, and delete employee names in the directory.

- IS Writable

  Members of the information services group can change, add, and delete employee names in the directory.

Repeating the Site Survey

Finally, you may need to run more than one site survey, particularly if your enterprise has offices in multiple cities or countries. You may find your informational needs to be so complex that you have to allow several different organizations to keep information at their local offices rather than at a single, centralized site. In this case, each office that keeps a master copy of information should run its own site survey. After the site survey 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.

Chapter 3

How to Design the Schema

The site survey you conducted in Chapter 2 generated information about the data you plan to store in your directory. Next, you must decide how to represent the data you store. Your directory schema describes the types of data you can store in your directory. During schema design, you map each data element to an LDAP attribute, and gather related elements into LDAP object classes. Well-designed schema help maintain the integrity of the data you store in your directory.

This chapter describes the directory schema and how to design schema for your unique needs. This chapter contains the following sections:

- Schema Design Process Overview
- iPlanet Standard Schema
- Customizing the Schema
- Maintaining Consistent Schema

For information on replicating schema, refer to “Schema Replication,” on page 119.

Schema Design Process Overview

During schema design, you select and define the object classes and attributes used to represent the entries stored by Directory Server. Schema design involves the following steps:

- Choosing predefined schema elements to meet as many of your needs as possible.
- Extending the standard Directory Server schema to define new elements to meet your remaining needs.
- Planning for schema maintenance.

It is best to use existing schema elements defined in the standard schema provided with Directory Server. Choosing standard schema elements helps ensure compatibility with directory-enabled applications. In addition, as the schema is based on the LDAP standard, you are assured it has been reviewed and agreed to by a wide number of directory users.

iPlanet Standard Schema

Your directory schema maintain the integrity of the data stored in your directory by imposing constraints on the size, range, and format of data values. You decide what types of entries your directory contains (people, devices, organizations, and so forth) and the attributes available to each entry.

The predefined schema included with Directory Server contains both the standard LDAP schema as well as additional application-specific schema to support the features of the server. While this schema meets most directory needs, you may need to extend it with new object classes and attributes to accommodate the unique needs of your directory. Refer to “Customizing the Schema” for information on extending the schema.

The following sections describe the format, standard attributes, and object classes included in the iPlanet standard schema.

Schema Format

Directory Server bases its schema format on version 3 of the LDAP protocol (LDAPv3). This protocol requires directory servers to publish their schemas through LDAP itself, allowing directory client applications to programmatically retrieve the schema and adapt their behavior based on it. The global set of schema for Directory Server can be found in the entry named `cn=schema`.

The Directory Server schema differs slightly from the LDAPv3 schema, as it uses its own proprietary object classes and attributes. In addition, it uses a private field in the schema entries called `X-ORIGIN`, which describes where the schema entry was defined originally. For example, if a schema entry is defined in the standard LDAPv3 schema, the `X-ORIGIN` field refers to RFC 2252. If the entry is defined by iPlanet for the Directory Server’s use, the `X-ORIGIN` field contains the value `iPlanet Directory Server`.

For example, the standard person object class appears in the schema as follows:

```
objectclasses: ( 2.5.6.6 NAME 'person' DESC 'Standard Person Object
 Class' SUP top MUST (objectlass $ sn $ cn) MAY (description $
 seealso $ telephoneNumber $ userPassword) X-ORIGIN 'RFC 2252' )
```

This schema entry states the object identifier, or OID, for the class (`2.5.6.6`), the name of the object class (`person`), a description of the class (`standard person`), then lists the required attributes (`objectclass`, `sn`, and `cn`) and the allowed attributes (`description`, `seealso`, `telephoneNumber`, and `userPassword`).

For more information about the LDAPv3 schema format, refer to the LDAPv3 Attribute Syntax Definitions document (RFC2252).

The differences between Directory Server 5.0 schema and Directory Server 4.x are described in the following table:

Directory Server 4.x Syntax	Equivalent Directory Server 5.0 Syntax
bin	Binary
ces	IA5String
cis	DirectoryString
dn	DN
int	INTEGER
tel	TelephoneNumber

In addition, Directory Server 5.0 supports the following syntaxes:

Syntax	Description
OctetString	Same behavior as Binary.
URI	Same behavior as IA5String.
Boolean	Boolean values have a value of either "TRUE" or "FALSE."
GeneralizedTime	Values in this syntax are encoded as printable strings. It is strongly recommended that GMT time be used. The time zone must be specified.

Syntax	Description
CountryString	A value in this syntax is encoded the same as a value of DirectoryString syntax. Note that this syntax is limited to values of exactly two printable string characters.

PostalAddress	Values in this syntax are encoded according to the following:
	`postal-address = dstring *("$" dstring)`
	In the above, each dstring component of a postal address value is encoded as a value of type DirectoryString syntax. Backslashes and dollar characters, if they occur in the component, are quoted. Many servers limit the postal address to six lines of up to thirty characters. For example:
	`1234 Main St.$Anytown,TX 12345$USA`

Standard Attributes

Attributes hold specific data elements such as a name or a fax number. Directory Server represents data as attribute-data pairs, a descriptive attribute associated with a specific piece of information. For example, the directory can store a piece of data such as a person's name in a pair with the standard attribute, in this case commonName (cn). So, an entry for a person named Babs Jensen has the following attribute-data pair:

```
cn: Babs Jensen
```

In fact, the entire entry is represented as a series of attribute-data pairs. The entire entry for Babs Jensen might appear as follows:

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

Notice that the entry for Babs contains multiple values for some of the attributes. The attribute `givenName` appears twice, each time with a unique value. The object classes that appear in this example are explained in the next section, “Standard Object Classes."

In the schema, each attribute definition contains the following information:

- A unique name
- An object identifier (OID) for the attribute
- A text description of the attribute
- The OID of the attribute syntax
- Indications of whether the attribute is single-valued or multi-valued, whether the attribute is for the directory’s own use, the origin of the attribute, and any additional matching rules associated with the attribute.

For example, the `cn` attribute definition appears in the schema as follows:

```
attributetypes: ( 2.5.4.3 NAME 'cn' DESC 'commonName Standard
 Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Standard Object Classes

Object classes are used to group related information. Typically, an object class represents a real object, such as a person or a fax machine. Before you can use an object class and its attributes in your directory, it must be identified in the schema. Your directory recognizes a standard list of object classes by default. See the *iPlanet Schema Reference* for more information.

Each directory entry belongs to one or more object classes. Once you place an object class identified in your schema 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.

Object class definitions contain the following information:

- A unique name
- An object identifier (OID) that names the object
- A set of mandatory attributes
- A set of allowed attributes

For an example of a standard object class as it appears in the schema, refer to “Schema Format,” on page 40.

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

Mapping Your Data to the Default Schema

The data you identified during your site survey, as described in “Performing a Site Survey,” on page 28, must be mapped to the existing directory default schema. This section describes how to view the existing default schema and provides a method for mapping your data to the appropriate existing schema elements.

If you find elements in your schema that do not match the existing default schema, you may need to create custom object classes and attributes. Refer to “Customizing the Schema,” on page 46 for more information.

Viewing the Default Directory Schema

The default directory schema is stored in the following place:
`/usr/iplanet/servers/slapd-`*serverID*`/config/schema`

This directory contains all of the common schema for the iPlanet products. The LDAPv3 standard user and organization schema can be found in the `00core.ldif` file. The configuration schema used by earlier version of the directory can be found in the `50ns-directory.ldif` file.

NOTE You should not modify the default directory schema.

For more information about each object class and attribute found in directory, refer to the *iPlanet Schema Reference*. For more information about the schema files and directory configuration attributes, refer to *iPlanet Directory Server Command and File Reference*.

Matching Data to Schema Elements

The data you identified in your site survey now needs to be mapped to the existing directory schema. This process involves the following steps:

- Identify the type of object the data describes.

  Select an object that best matches the data described in your site survey. Sometimes, a piece of data can describe multiple objects. You need to determine if the difference needs to be noted in your directory schema. For example, a telephone number can describe an employee's telephone number and a conference room's telephone number. It is up to you to determine if these different sorts of data need to be considered different object in your directory schema.

- Select a similar object class from the default schema.

  It is best to use the common object classes, such as groups, people, and organizations.

- Select a similar attribute from the matching object class.

  Select an attribute from within the matching object class that best matches the piece of data you identified in your site survey.

- Identify the unmatched data from your site survey.

  If there are some pieces of data that do not match the object classes and attributes defined by the default directory schema, you will need to customize the schema. See "Customizing the Schema," on page 46 for more information.

For example, the following table maps directory schema elements to the data identified during the site survey in Chapter 2:

Table 3-1 Data Mapped to Default Directory Schema

Data	Owner	Object Class	Attribute
Employee name	HR	person	cn(commonName)
User password	IS	person	userPassword
Home phone number	HR	inetOrgPerson	homePhone
Employee location	IS	inetOrgPerson	localityName
Office phone number	Facilities	person	telephoneNumber

In the table, the employee name describes a person. In the default directory schema, we found the `person` object class, which inherits from the `top` object class. This object class allows several attributes, one of which is the `cn` or `commonName` attribute, which describes the full name of the person. This attribute makes the best match for containing the employee name data.

The user password also describes an aspect of the `person` object. In the list of allowed attributes for the person object, we find `userPassword`.

The home phone number describes an aspect of a person, however we do not find an appropriate attribute in the list associated with the `person` object class. Analyzing the home phone number more specifically, we can say it describes an aspect of a person in an organization’s enterprise network. This object corresponds to the `inetOrgPerson` object class in the directory schema. The `inetOrgPerson` object class inherits from the `organizationPerson` object class, which in turn inherits from the `person` object class. Among the `inetOrgPerson` object’s allowed attributes, we locate the `homePhone` attribute, which is appropriate for containing the employee’s home telephone number.

Customizing the Schema

You can extend the standard schema if it proves to be too limited for your directory needs. To help you extend your schema, the iPlanet Directory Server includes a schema management tool. For more information, see the *iPlanet Directory Server Administrator’s Guide*.

Keep the following rules in mind when customizing your schema:

- Reuse existing schema elements whenever possible.
- Minimize the number of mandatory attributes you define for each object class.
- Do not define more than one object class or attribute for the same purpose.
- Keep the schema as simple as possible.
- Do not modify any existing definitions of attributes or object classes.

NOTE When customizing the schema you should never delete or replace the standard schema. Doing so can lead to compatibility problems with other directories or other LDAP client applications.

Your custom object classes and attributes are defined in the following file:

`/usr/iplanet/servers/slapd-`*serverID*`/config/schema/99user.ldif`

The following sections describe customizing the directory schema in more detail:

- “When to Extend Your Schema,” on page 47

When to Extend Your Schema

While the object classes and attributes supplied with the Directory Server should meet most of your needs, you may find that a given object class does not allow you to store specialized information about your organization. Also, you may need to extend your schema to support the object classes and attributes required by an LDAP-enabled application’s unique data needs.

Getting and Assigning Object Identifiers

Each LDAP object class or attribute must be assigned a unique name and object identifier (OID). When you define a schema, you need an OID unique to your organization. One OID is enough to meet all of your schema needs. You simply add another level of hierarchy to create new branches for your attributes and object classes. Getting and assigning OIDs in your schema involves the following steps:

- Obtain an OID for your organization from the Internet Assigned Numbers Authority (IANA) or a national organization.

  In some countries, corporations already have OIDs assigned to them. If your organization does not already have an OID, one can be obtained from IANA. For more information, go to the IANA website at http://www.iana.org/cgi-bin/enterprise.pl.

- Create an OID registry so you can track OID assignments.

  An OID registry is a list you maintain that gives the OIDs and descriptions of the OIDs used in your directory schema. This ensures that no OID is ever use for more than one purpose. You should then publish your OID registry with your schema.

- Create branches in the OID tree to accommodate schema elements.

  Create at least two branches under the OID branch or your directory schema, using *OID*`.1` for attributes and *OID*`.2` for object classes. If you want to define your own matching rules or controls, you can add new branches as needed (*OID*`.3`, for example).

Naming Attribute and Object Classes

When creating names for new attribute and object classes, make the name as meaningful as possible. This makes your schema easier to use for Directory Server administrators.

Avoid naming collisions between your schema elements and existing schema elements by including a unique prefix on all of your elements. For example, the Siroe company might add the prefix `siroe` before each of their custom schema elements. They might add a special object class called `siroePerson` to identify Siroe employees in their directory.

Strategies for Defining New Object Classes

There are two ways you can create new object classes:

- You can create many new object classes, one for each object class structure to which you want to add an attribute.
- You can create a single object class that supports all of the attributes that you create for your directory. You create this kind of an object class by defining it to be an AUXILIARY kind of object class.

You may find it easiest to mix the two methods.

For example, suppose your site wants to create the attributes `siroeDateOfBirth`, `siroePreferredOS`, `siroeBuildingFloor`, and `siroeVicePresident`. You can create several object classes that allow some subset of these attributes. You might create an object class called `siroePerson` and have it allow `siroeDateOfBirth` and `siroePreferredOS`. The parent of `siroePerson` would be `inetOrgPerson`. You might then create an object class called `siroeOrganization` and have it allow `siroeBuildingFloor` and `siroeVicePresident`. The parent of `siroeOrganization` would be the `organization` object class.

Your new object classes would appear in LDAPv3 schema format as follows:

```
objectclasses: ( 2.16.840.1.17370.999.1.2.3 NAME 'siroePerson'  DESC
'Siroe Person Object Class' SUP inetorgPerson MAY (siroeDateOfBirth
 $ siroePreferredOS) )
```

```
objectclasses: ( 2.16.840.1.17370.999.1.2.4 NAME
 'siroeOrganization' DESC 'Organization Object Class' SUP
 organization MAY (siroeBuildingFloor $ siroeVicePresident) )
```

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 these attributes. The single object class would appear as follows:

```
objectclasses: (2.16.840.1.17370.999.1.2.5 NAME 'siroeEntry' DESC
 'Standard Entry Object Class' SUP top AUXILIARY MAY
 (siroeDateOfBirth $ siroePreferredOS $ siroeBuildingFloor $
 siroeVicePresident) )
```

The new `siroeEntry` object class is marked AUXILIARY, meaning that it can be used with any entry regardless of its structural object class.

NOTE The OID of the new object classes in the example is based on the iPlanet OID prefix. To create your own new object classes, you must get your own OID. For more information, refer to “Getting and Assigning Object Identifiers,” on page 47.

Choose the strategy for defining new object classes that works for you. Consider the following when deciding how to implement new object classes:

- Multiple object classes result in more schema elements to create and maintain.

  Generally, the number of elements remains small and need little maintenance. However, you may find it easier to use a single object class if you plan to add more than two or three object classes to your schema.

- Multiple object classes require a more careful and rigid data design.

  Rigid data design 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.

- Single object classes simplify data design when 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. You may want to create only a single object class to allow this attribute.

- Avoid required attributes for new object classes.

Requiring attributes can make your schema inflexible. When create a new object class, allow rather than require attributes.

After defining a new object class, you need to decide what attributes it allows and requires and from what object class(es) it inherits.

Strategies for Defining New Attributes

Add new attributes and new object classes when the existing object classes do not support all of the information you need to store in a directory entry.

Try to use standard attributes whenever possible. Search the attributes that already exist in the default directory schema and use them in association with a new object class. Create a new attribute if you cannot find a match in the default directory schema.

For example, you may find that you want to store more information on a person entry than the `person`, `organizationalPerson`, or `inetOrgPerson` object classes support. If you want to store the birth dates in your directory, no attribute exists within the standard iPlanet Directory Server schema. You can choose to create a new attribute called `dateOfBirth` and allow this attribute to be used on entries representing people by defining a new auxiliary class, `siroePerson`, which allows this attribute.

Deleting Schema Elements

Do not delete the schema elements shipped with Directory Server. Unused schema elements represent no operational or administrative overhead. However, by deleting parts of the standard LDAP schema you may run into compatibility problems with future installations of Directory Server and other directory-enabled applications.

However, if you extend the schema and find you do not use the new elements, feel free to delete the elements you don’t use. Before removing the object class definitions from the schema, you need to modify each entry using the object class. Otherwise, if you remove the definition first, you might not be able to modify the entries that use the object class afterwards. Schema checks on modified entries will also fail unless you remove the unknown object class values from the entry.

Creating Custom Schema Files

You can create custom schema files other than the 99user.ldif file provided with Directory Server. However, your custom schema files should not be numerically or alphabetically higher than 99user.ldif or the server could experience problems.

The 99user.ldif file contains attributes with the an `X-ORIGIN` of `'user defined'`. If you create a schema file called 99zzz.ldif, the next time you update the schema using LDAP or the Directory Server Console, all of the attributes with an `X-ORIGIN` value of `'user defined'` will be written to 99zzz.ldif. The directory writes them to 99zzz.ldif because the directory uses the highest sequenced file (numerically, then alphabetically) for its internal schema management. The result is two LDIF files that contain duplicate information and some information in the 99zzz.ldif file might be erased.

When naming custom schema files, name them as follows:

`[00-99]`*`yourname`*`.ldif`

The directory loads these schema files in numerical order, followed by alphabetical order. For this reason, you should use a number scheme that is higher than any directory standard schema defined. For example, Siroe Corporation creates a new schema file named `60siroecorp.ldif`. If you name your schema file something lower than the standard schema files, the server may encounter errors when loading the schema. In addition, all standard attributes and object classes will be loaded only after your custom schema elements have been loaded.

You should not use `'user defined'` in the `X-ORIGIN` field of your custom schema files as `'user defined'` is used internally by the directory when schema is added over LDAP. Use something more descriptive, such as `'Siroe Corporation defined'`.

After you have created custom schema files, you can either:

- Manually copy these custom schema files to all of your servers, which requires a restart of each server.
- Allow the replication process to replicate this information to each of the consumers for you.

If you do not copy these custom schema files to all of your servers, the schema information will only be replicated to the consumer when changes are made to the schema on the supplier using LDAP or the Directory Server Console.

When the schema definitions are then replicated to a consumer server where they do not already exist, they will be stored in the 99user.ldif file. The directory does not track where schema definitions are stored. Storing schema elements in the 99user.ldif file of consumers does not create a problem as long as you maintain your schema on the master server only.

If you copy your custom schema files to each server, when changes are made to the schema files, they must be copied again to each server. If you do not copy them again, it is possible the changes will be replicated and stored in the 99user.ldif file on the consumer. Having the changes in the 99user.ldif file may make schema management difficult, as some attributes will appear in two separate schema files on a consumer, once in the original custom schema file you copied from the supplier and again in the 99user.ldif file after replication.

For more information about replicating schema, see "Schema Replication," on page 119.

Custom Schema Best Practices

Consider the following points when creating custom schema elements:

- If you manage your schema using LDAP or the Directory Server Console, add all of your schema definitions to the 99user.ldif file to avoid possible duplication of schema elements in multiple files. Schema elements added and updated via LDAP are automatically written to the 99user.ldif file.

NOTE If you define custom schema files, for example 60siroecorp.ldif, and then update these schema elements using LDAP, the new definitions will be written to the 99user.ldif file and not to your custom schema file, thus overriding your original custom schema definition. For example, changes to 60siroecorp.ldif will be overwritten by the definitions stored in 99user.ldif.

- If adding schema elements to the 99user.ldif manually, always use an `X-ORIGIN` of value `'user defined'`. If you use something other than `'user defined'`, when the server loads the schema from the 99user.ldif file, it will add the `'user defined'` value to the `X-ORIGIN` portion of the definition in addition to what you have already specified for the `X-ORIGIN`. The result is that attributes that are not `'user defined'` will appear in the read-only section of the Directory Server Console, and you will not be able to use the Console to edit object classes that contain an `X-ORIGIN` other than `'user defined'`.

Using an X-ORIGIN of value 'user defined' ensures that schema definitions in the 99user.ldif file are not removed from the file by the directory. The directory does not remove them because it relies on an X-ORIGIN of'user defined' to tell it what elements should reside in the 99user.ldif file.

For example, you create a schema entry manually in 99user.ldif as follows:

```
attributetypes: ( siroeContact-oid NAME 'siroeContact' DESC
 'Siroe Corporate contact' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 X-ORIGIN 'Siroe defined')
```

After the directory loads the schema entry, it appears as follows:

```
attributetypes: ( siroeContact-oid NAME 'siroeContact' DESC
 'Siroe Corporate contact' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 X-ORIGIN ('Siroe defined' 'user defined') )
```

- When adding new schema elements, all attributes need to be defined before they can be used in an object class. You can define attributes and object classes in the same schema file.

- Each custom attribute or object class you create should be defined in only one schema file. This prevents the server from overriding any previous definitions when it loads the most recently created schema (as the server loads the schema in numerical order first, then alphabetical order).

Maintaining Consistent Schema

A consistent schema within Directory Server aids LDAP client applications in locating directory entries. If you use an inconsistent schema, then it becomes very difficult to locate information in your directory tree efficiently.

Inconsistent schema use different attributes or formats to store the same information. You can maintain schema consistency in the following ways:

- Use schema checking to ensure attributes and object classes conform to the schema rules.

- Select and apply a consistent data format.

The following sections describe in detail how to maintain consistency within your schema.

Schema Checking

Schema checking ensures that all new or modified directory entries conform to the schema rules. When the rules are violated, the directory rejects the requested change.

NOTE Schema checking checks only that the proper attributes are present. It does not verify whether attribute values are in the correct syntax for the attribute.

By default, the directory enables schema checking. We do not recommend turning it off. For information on turning schema checking on and off, see the *iPlanet Directory Server Administrator's Guide.*

With schema checking on, you must be attentive to required and allowed attributes as defined by the object classes. Object class definitions usually contain at least one required attribute, and one or more optional attributes. 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 Directory Server returns an object class violation message.

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`.

Selecting Consistent Data Formats

LDAP schema allow you to place any data that you want on any attribute value. However, it is important to store data consistently in your directory tree by selecting a format appropriate for your LDAP client applications and directory users.

With the LDAP protocol and iPlanet Directory Server, you must represent data in the data formats specified in RFC 2252.

In addition, the correct LDAP format for telephone numbers is defined in the following ITU-T Recommendations documents:

- ITU-T Recommendation E.123.

  Notation for national and international telephone numbers.
- ITU-T Recommendation E.163.

  Numbering plan for the international telephone services.

For example, a US phone number would be formatted as follows:

```
+1 555 222 1717
```

The `postalAddress` attribute expects an attribute value in the form of a multiline string that uses dollar signs ($) as line delimiters. A properly formatted directory entry appears as follows:

```
postalAddress: 1206 Directory Drive$Pleasant View, MN$34200
```

Maintaining Consistency in Replicated Schema

When you make changes to your directory schema, the change is recorded in the change log. During replication, the change log is scanned for changes, and any changes made are replicated. Maintaining consistency in replicated schema allows replication to continue smoothly. Consider the following points for maintaining consistent schema in a replicated environment:

- Do not modify the schema on a read-only replica.

  Modifying the schema on a read-only replica introduces an inconsistency in your schema and causes replication to fail.
- You cannot create two attributes with the same name that use different syntaxes.

  If you create an attribute in a read-write replica that has the same name as an attribute on the master replica, but has a different syntax from the attribute on the master, replication will fail.

Other Schema Resources

Refer to the following links for more information about standard LDAPv3 schema:

- Internet Engineering Task Force (IETF)
  http://www.ietf.org/

- *Understanding and Deploying LDAP Directory Services.*
  T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999.
- RFC 2252: LDAPv3 Attribute Syntax Definitions
  http://www.ietf.org/rfc/rfc2252.txt
- RFC 2256: Summary of the X.500 User Schema for Use with LDAPv3
  http://www.ietf.org/rfc/rfc2256.txt
- RFC 2251: Lightweight Directory Access Protocol (v3)
  http://www.ietf.org/rfc/rfc2251.txt

Chapter 4

Designing the Directory Tree

Your directory tree provides a way to refer to the data stored by your directory. The types of information you store in your directory, the physical nature of your enterprise, the applications you use with your directory, and the types of replication you use shape the design of your directory tree.

This chapter outlines the steps for designing your own directory tree. It includes the following sections:

- Introduction to the Directory Tree
- Designing Your Directory Tree
- Grouping Directory Entries
- Directory Tree Design Examples

Introduction to the Directory Tree

Your directory tree provides a means for your directory data to be named and referred to by client applications. Your directory tree interacts closely with other design decisions, including the choices available to you when you decide on how to distribute, replicate, or control access to your directory data. Taking the time to design your directory tree well before deployment saves headaches later if you find it inadequate after you have launched your directory.

A well-designed directory tree provides the following:

- Simplified directory data maintenance
- Flexibility in creating replication policies and access controls
- Support for the applications using your directory
- Simplified directory navigation for directory users

The structure of your directory tree follows the hierarchical LDAP model. Your directory tree provides a way to organize your data, for example, by group, by people, or by place. It also determines how you partition data across multiple servers. For example, each database needs data to be partitioned at the suffix level. Without the proper directory tree structure, you may not be able to spread your data across multiple servers as you would like.

In addition, replication is constrained by what sort of directory tree structure you use. You must carefully define partitions for replication to work. If you want to replicate only portions of your directory tree, you need to take that into account during the design process. If you plan to use access controls on branch points, that is also a consideration in your directory tree design.

Designing Your Directory Tree

This section guides you through the major decisions you make during the directory tree design process. The directory tree design process involves the following steps:

- Choosing a suffix to contain your data
- Determining the hierarchical relationship among data entries
- Naming the entries in your directory tree hierarchy

The following sections describe the directory tree design process in more detail.

Choosing a Suffix

The suffix is the name of the entry at the root of your tree, below which you store your directory data. Your directory can contain more than one suffix. You may choose to use multiple suffixes if you have two or more directory trees of information that do not have a natural common root.

By default, the standard iPlanet Directory Server deployment contains multiple suffixes, one for storing data and the others for data needed by internal directory operations (such as configuration information and your directory schema). For more information on these standard directory suffixes, see the *iPlanet Directory Server Administrator’s Guide*.

Suffix Naming Conventions

All entries in your directory should be located below a common base entry, the root suffix. Consider the following recommendations for naming the root directory suffix:

- Globally unique
- Static, so it rarely changes, if ever
- Short, so that entries beneath it are easier to read on screen
- Easy for a person to type and remember

In a single enterprise environment, choose a directory suffix that aligns with a DNS name or Internet domain name of your enterprise. For example, if your enterprise owns the domain name of `siroe.com`, then you should use a directory suffix of:

```
dc=siroe,dc=com
```

The `dc` (`domainComponent`) attribute represents your suffix by breaking your domain name into its component parts.

Normally, you can use any attribute that you like to name your root suffix. However, for a hosting organization, we recommend that the root suffix contain only the following attributes:

- `c (countryName)`

  Contains the two-digit code representing the country name, as defined by ISO.

- `l (localityName)`

  Identifies the county, city, or other geographical area where the entry is located or which is associated with the entry.

- `st`

  Identifies the state or province where the entry resides.

- `o (organizationName)`

  Identifies the name of the organization to which the entry belongs.

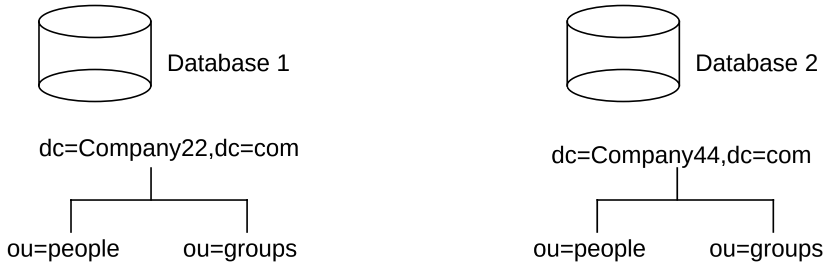
The presence of these attributes allows for interoperability with subscriber applications. For example, a hosting organization might use these attributes to create the following root suffix for one of its clients, Company22:

```
o=Company222,st=Washington,c=US
```

Using an organization name followed by a country designation is typical of the X.500 naming convention for suffixes.

Naming Multiple Suffixes

Each suffix that you use with your directory is a unique directory tree. There are several ways that you can include multiple trees in your directory. The first is to create multiple directory trees stored in separate databases served by Directory Server. For example, you could create separate suffixes for the Company22 and the Company44 and store them in separate databases as follows:



The databases could be stored on a single server or multiple servers depending upon resource constraints.

Creating Your Directory Tree Structure

You need to decide whether you use a flat or hierarchical tree structure. As a general rule, strive to make your directory tree as flat as possible. However, a certain amount of hierarchy can be important later when you partition data across multiple databases, prepare replication, and set access controls.

The structure of your tree involves the following steps and considerations:

- "Branching Your Directory," on page 60
- "Identifying Branch Points," on page 62
- "Replication Considerations," on page 64
- "Access Control Considerations," on page 66

Branching Your Directory

Design your hierarchy to avoid problematic name changes. The flatter a namespace is, the less likely the names are to change. The likelihood of a name changing is roughly proportional to the number of components in the name that can potentially change. The more hierarchical the directory tree, the more components in the names, and the more likely the names are to change.

Following are some guidelines for designing your directory tree hierarchy:

- Branch your tree to represent only the largest organizational subdivisions in your enterprise.

  Any such branch points should be limited to divisions (Corporate Information Services, Customer Support, Sales and Professional Services, and so forth). Make sure that divisions you use to branch your directory tree are stable; do not perform this kind of branching if your enterprise reorganizes frequently.

- Use functional or generic names rather than actual organizational names for 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 for that function instead of branching based along divisional lines.

  For example, even if you have multiple marketing organizations, each of which is responsible for a specific product line, create a single Marketing subtree. All marketing entries then belong to that tree.

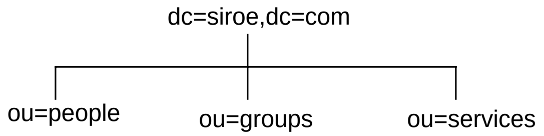
Following are specific guidelines for the enterprise and hosting environment.

Branching in an Enterprise Environment

Name changes can be avoided if you base your directory tree structure on information that is not likely to change. For example, base the structure on types of objects in the tree rather than organizations. Some of the objects you might use to define your structure are:

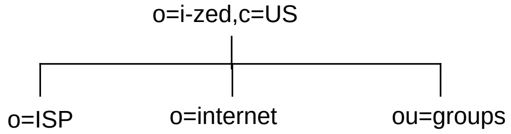
- `ou=people`
- `ou=groups`
- `ou=contracts`
- `ou=employees`
- `ou=services`

A directory tree organized using these objects might appear as follows:



Branching in a Hosting Environment

For a hosting environment, create a tree that contains two entries of the object class `organization(o)` and one entry of the `organizationalUnit(ou)` object class beneath the root suffix. For example, the ISP I-Zed branches their directory as follows:

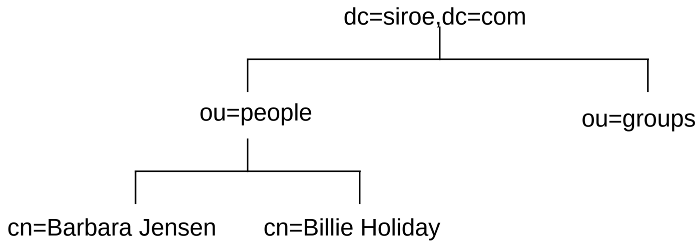


Identifying Branch Points

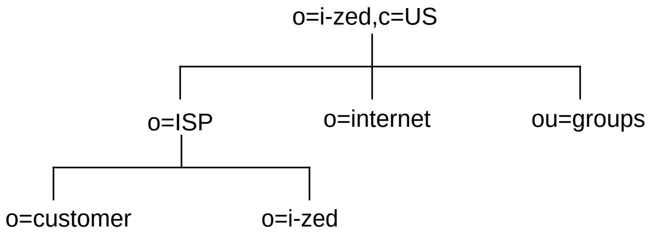
As you decide how to branch your directory tree, you will need to decide what attributes you will use to identify the branch points. Remember that a DN is a unique string composed of attribute-data pairs. For example, the DN of an entry for Barbara Jensen, an employee of Siroe Corporation, appears as follows:

```
cn=Barbara Jensen,ou=people,dc=siroe,dc=com
```

Each attribute-data pair represents a branch point in your directory tree. For example, the directory tree for the enterprise Siroe Corporation appears as follows:



The directory tree for I-Zed, an internet host, appears as follows:



Beneath the root suffix entry, `o=i-zed,c=US`, the tree is split into three branches. The ISP branch contains customer data and internal information for I-Zed. The internet branch is the domain tree. The groups branch contains information about the administrative groups.

There are some points to consider when choosing attributes for your branch points:

- Be consistent.

  Some LDAP client applications may be confused if the distinguished name (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 4-1).

  Using traditional attributes increases the likelihood of retaining compatibility with third-party LDAP client applications. Using the traditional attributes also means that they will be known to the default directory schema, which makes it easier to build entries for the branch DN.

Table 4-1 Traditional DN Branch Point Attributes

Attribute Name	**Definition**
`c`	A country name.
`o`	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 “Suffix Naming Conventions,” on page 59.

Table 4-1 Traditional DN Branch Point Attributes *(Continued)*

Attribute Name	Definition
`ou`	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`	A state or province name.
`l`	A locality, such as a city, country, office, or facility name.
`dc`	A domain component as discussed in “Suffix Naming Conventions,” on page 59.

NOTE A common mistake is to assume that you search your directory based on the attributes used in the distinguished name. However, the distinguished name is only a unique identifier for the directory entry and cannot be searched against.

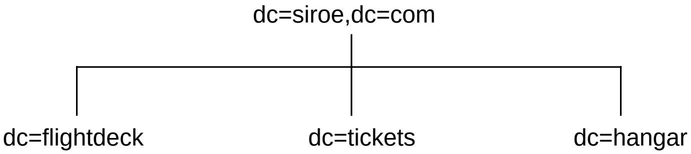
Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is `cn=Babs Jensen,ou=People,dc=siroe,dc=com`, then a search for `dc=siroe` will not match that entry unless you have explicitly put `dc: sun` as an attribute in that entry.

Replication Considerations

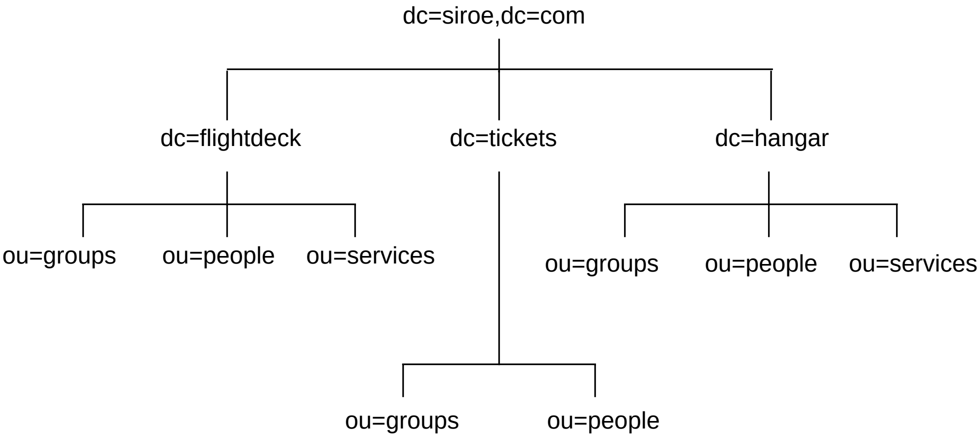
During directory tree design, consider which entries you are replicating. A natural way to describe a set of entries to be replicated is to specify the distinguished name (DN) at the top of a subtree and replicate all entries below it. This subtree also corresponds to a database, a directory partition containing a portion of the directory data.

For example, in an enterprise environment you can organize your directory tree so that it corresponds to the network names in your enterprise. Network names tend to not change, so the directory tree structure will be stable. Further, using network names to create the top level branches of your directory tree is useful when you use replication to tie together different directory servers.

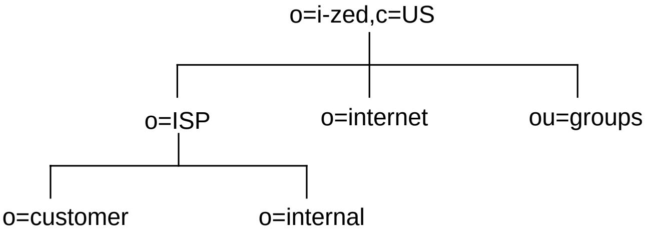
For example, Siroe Corporation has three primary networks known as `flightdeck.siroe.com`, `tickets.siroe.com`, and `hanger.siroe.com`. They initially branch their directory tree as follows:



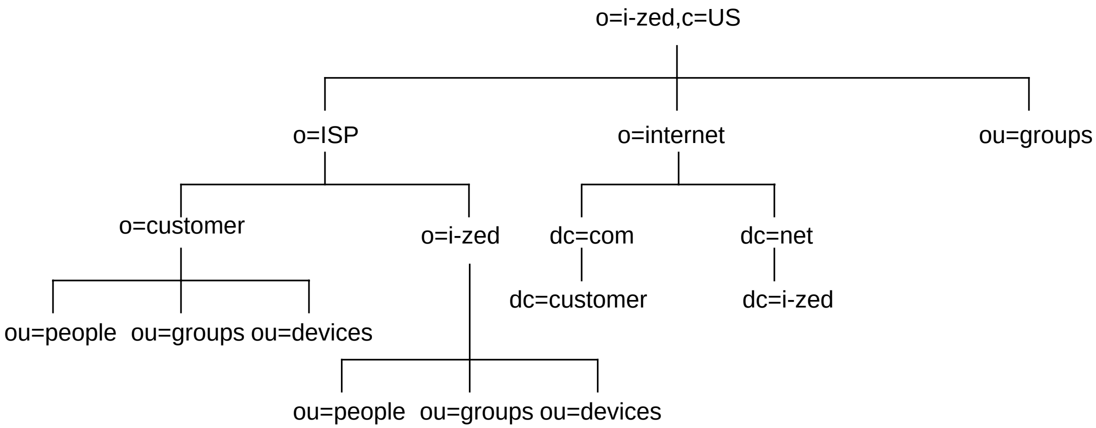
After creating the initial structure of the tree, they create additional branches as follows:



As another example, the ISP i-zed.com branches their directory as follows:



After creating the initial structure of their directory tree, they create additional branches as follows:



Both the enterprise and the hosting organization design their data hierarchies based on information that is not likely to change often.

Access Control Considerations

Introducing hierarchy into your directory tree can be used to enable certain types of access control. As with replication, it is easier to group together similar entries and then administer them from a single branch.

You can also enable the distribution of administration through a hierarchical directory tree. For example, if you want give an administrator from the marketing department access to the marketing entries and an administrator from the sales department access to the sales entries, you can do so through your directory tree design.

You can set access controls based on the directory content rather than the directory tree. The ACI filtered target mechanism lets you define a single access control rule stating that a directory entry has access to all entries containing a particular attribute value. For example, you could set an ACI filter that gives the sales administrator access to all the entries containing the attribute `ou=Sales`.

However, ACI filters can be difficult to manage. You must decide which method of access control is best suited to your directory: organizational branching in your directory tree hierarchy, ACI filters, or a combination of the two.

Naming Entries

After designing the hierarchy of your directory tree, you need to decide which attributes to use when naming the entries within the structure. Generally, names are created by choosing one or more of the attribute values to form a relative distinguished name (RDN). The RDN is the left-most DN attribute value. The attributes you use depend on the type of entry you are naming.

Your entry names should adhere to the following rules:

- The attribute you select for naming should be unlikely to change.
- The name must be unique across your directory.

  A unique name ensures that a DN can refer to at most one entry in your directory.

When creating entries, define the RDN within the entry. By defining at least the RDN within the entry, you can locate the entry more easily. This is because searches are not performed against the actual DN but rather the attribute values stored in the entry itself.

Attribute names have a meaning, so try to use the attribute name that matches the type of entry it represents. For example, do not use `l` (`locality`) to represent an organization, or `c` (`country`) to represent an organizational unit.

The following sections provide tips on naming entries:

- Naming Person Entries
- Naming Group Entries
- Naming Organization Entries
- Naming Other Kinds of Entries

Naming Person Entries

The person entry’s name, the DN, must be unique. Traditionally, distinguished names use the `commonName`, or `cn`, attribute to name their person entries. That is, an entry for a person named Babs Jensen might have the distinguished name of:

```
cn=Babs Jensen,dc=siroe,dc=com
```

While allowing you to instantly recognize the person associated with the entry, it might not be unique in an include people with identical names. This quickly leads to a problem known as DN name collisions, multiple entries with the same distinguished name.

You can avoid common name collisions by adding a unique identifier to the common name. For example:

```
cn=Babs Jensen+employeeNumber=23,dc=siroe,dc=com
```

However, this can lead to awkward common names for large directories and can be difficult to maintain.

A better method is to identify your person entries with some attribute other than `cn`. Consider using one of the following attributes:

- `uid`

  Use the `uid` (`userID`) attribute to specify some unique value of the person. Possibilities include a user login ID or an employee number. A subscriber in a hosting environment should be identified by the `uid` attribute.

- `mail`

  Use the `mail` attribute to contain the value for the person's email address. This option can lead to awkward DNs that include duplicate attribute values (for example: `mail=bjensen@siroe.com, dc=siroe,dc=com`), so you should use this option only if you cannot find some unique value that you can use with the `uid` attribute. For example, you would use the `mail` attribute instead of the `uid` attribute if your enterprise does not assign employee numbers or user IDs for temporary or contract employees.

- `employeeNumber`

  For employees of the `inetOrgPerson` object class, consider using an employer assigned attribute value such as `employeeNumber`.

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 readable. For example, `uid=bjensen, dc=siroe,dc=com` is preferable to `uid=b12r56A, dc=siroe,dc=com` because recognizable DNs simplify some directory tasks, such as changing directory entries based on their distinguished names. Also, some directory client applications assume that the `uid` and `cn` attributes use human-readable names.

Considerations for Person Entries in a Hosted Environment

If a person is a subscriber to a service, the entry should be of object class `inetUser` and the entry should contain the `uid` attribute. The attribute must be unique within a customer subtree.

If a person is part of the hosting organization, represent them as an `inetOrgPerson` with the `nsManagedPerson` object class.

Placing Person Entries in the DIT

Here are some guidelines for placing people entries in your directory tree:

- People in an enterprise should be located in the directory tree below the organization's entry.
- Subscribers to a hosting organization need to be below the `ou=people` branch for the hosted organization.

Naming Group Entries

There are four main ways to represent a group:

- A static group

  The entry for this type of group uses the `groupOfNames` or `groupOfUniqueNames` object class, which contains values naming the members of the group. Static groups are suitable for groups with few members, such as the group of directory administrators. Static groups are not suitable for groups with thousands of members.

  Static group entries must contain a `uniqueMember` attribute value, because `uniqueMember` is a mandatory attribute of the `groupOfUniqueNames` object. This object class requires the `cn` attribute, which can be used to form the DN of the group entry.

- A member-based group

  This type of group uses a `memberOf` attribute in the entry of each group member.

- A dynamic group

  This type of group uses an entry representing the group with a search filter and subtree. Entries matching the filter are members of the group.

- A Directory Server role

  Roles are a new feature of Directory Server that unify the static and dynamic group concept. Refer to "About Roles," on page 71 for more information.

In a deployment containing hosted organizations, we recommend using the `groupOfUniqueNames` object class to contain the values naming the members of groups used in directory administration. In a hosted organization, we also recommend that group entries used for directory administration are located under the `ou=Groups` branch.

Naming Organization Entries

The organization entry name, like other entry names, must be unique. Using the legal name of the organization along with other attribute values helps ensure the name is unique. For example, you might name an organization entry as follows:

```
o=Company22+st=Washington,o=ISP,c=US
```

You can also use trademarks, however they are not guaranteed to be unique.

In a hosting environment, you need to include the following attributes in the organization's entry:

- `o (organizationName)`
- `objectClass` with values of `top`, `organization`, and `nsManagedDomain`

Naming Other Kinds of Entries

Your directory will contain entries that represent many things, such as localities, states, countries, devices, servers, network information, and other kinds of data.

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,dc=siroe,dc=com
```

However, sometimes you need to name an entry whose object class does not support the `commonName` attribute. Instead, use an attribute that is supported by the entry's object class.

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. However, a correspondence between the DN attributes and attributes used by the entry simplifies administration of your directory tree.

Grouping Directory Entries

Once you have created entries, you can group them for ease of administration. The Directory Server supports several methods for grouping entries and sharing attributes between entries:

- Using roles
- Using class of service

The following sections describe each of these mechanisms in more detail.

About Roles

Roles are a new entry grouping mechanism. Your directory tree organizes information hierarchically. This hierarchy is a grouping mechanism, though it is not suited for short-lived, changing organizations. Roles provide another grouping mechanism for more temporary organizational structures.

Roles unify static and dynamic groups. You use static groups to create a group entry that contains a list of members. Dynamic groups allow you to filter entries that contain a particular attribute and include them in a single group.

Each entry assigned to a role contains the `nsRole` attribute, a computed attribute that specifies all of the roles an entry belongs to. A client application can check role membership by searching the `nsRole` attribute, which is computed by the directory and therefore always up-to-date.

Roles are designed to be more efficient and easier to use for applications. For example, applications can locate the roles of an entry, rather than select a group and browse the members list.

You can use roles to do the following:

- Enumerate the members of the role.

  Having an enumerated list of role members can be useful for resolving queries for group members quickly.

- Determine whether a given entry possesses a particular role.

  Knowing the roles possessed by an entry can help you determine whether the entry possesses the target role.

- Enumerate all the roles possessed by a given entry.
- Assign a particular role to a given entry.
- Remove a particular role from a given entry.

Each role has *members*, entries that possess the role. You can specify members either explicitly (meaning each entry contains an attribute associating it with a role) or dynamically (by creating a filter that assigns entries to roles depending upon an attribute contained by the entry). How you specify role membership depends upon the type of role you are using. There are three types of roles:

- Managed roles.

  A managed role allows you to create an explicit enumerated list of members. Managed roles are added to entries using the `nsRoleDN` attribute.

- Filtered roles.

  A filtered role allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

- Nested roles.

  A nested role allows you to create roles that contain other roles. You specify the roles nested within it using the `nsRoleDN` attribute.

Deciding Between Roles and Groups

Both methods of grouping entries have advantages and disadvantages. Roles reduce client-side complexity at the cost of increased server complexity. With roles, the client application can check role membership by searching the `nsRole` attribute. From the client application point of view, the method for checking membership is uniform and is performed on the server side.

Dynamic groups, from an application point of view, offer no support from the server to provide a list of group members. Instead, the application retrieves the group definitions and then runs the filter. For static groups, the application must make sure the user is part of a particular `UniqueMember` attribute value. The method for determining group membership is not uniform.

You can use managed roles to do everything you would normally do with static groups. You can filter group members using filtered roles as you used to do with dynamic groups.

While roles are easier to use, more flexible, and reduce client complexity, they do so at the cost of increased server complexity. Determining role membership is more resource intensive because the server does the work for the client application.

About Class of Service

A class of service (CoS) allows you to share attributes between entries in a way that is invisible to applications. With CoS, some attribute values may not be stored with the entry itself. Instead, they are generated by class of service logic as the entry is sent to the client application.

For example, your directory contains thousands of entries that all share the common attribute `facsimileTelephoneNumber`. Traditionally, to change the fax number, you would need to update each entry individually, a large job for administrators that runs the risk of not updating all entries. With CoS, you can generate the attribute dynamically. The `facsimileTelephoneNumber` attribute is stored in one place, and each entry points to that place to give a value to their fax number attribute. For the application, these attributes appear just like all other attributes, despite that they are not actually stored on the entries themselves.

Each CoS is comprised of the following entries in your directory:

- CoS Definition Entry.

  The CoS definition entry identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

- Template Entry.

  The template entry contains a list of the shared attribute values. Changes to the template entry attribute values are automatically applied to all the entries sharing the attribute.

The CoS definition entry and the template entry interact to provide attribute values to their *target entries*, the entries within their scope. The value they provide depends upon the following:

- The entry’s DN (different portions of the directory tree might contain different CoS).

- A service class attribute value stored with the entry.

  The absence of a service class attribute can imply a specific default CoS.

- The attribute value stored in the CoS template entry.

  Each CoS template entry supplies the attribute value for a particular CoS.

- The object class of the entry.

  CoS attribute values are generated only when an entry contains an object class allowing the attribute when schema checking is turned on, otherwise all attribute values are generated.

- The attribute stored in some particular entry in the directory tree.

You can use different types of CoS depending upon how you want the value of your dynamic attributes to be generated. There are three types of CoS:

- Pointer CoS.

  A pointer CoS identifies the template entry using the template DN only. There may be only one template DN for each pointer CoS. A pointer CoS applies to all entries within the scope of the template entry.

- Indirect CoS.

An indirect CoS identifies the template entry using the value of one of the target entry's attributes. The target entry's attribute must contain the DN of an existing entry.

- Classic CoS.

  A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes. Classic Cos can have multiple template entries, including a default CoS template to be applied to those entries that do not belong to any other CoS template.

Roles and the Classic CoS can be used together to provide role-based attributes. These attributes appear on an entry because it possesses a particular role with an associated class of service template. For example, you could use a role-based attribute to set the server look through limit on an role-by-role basis.

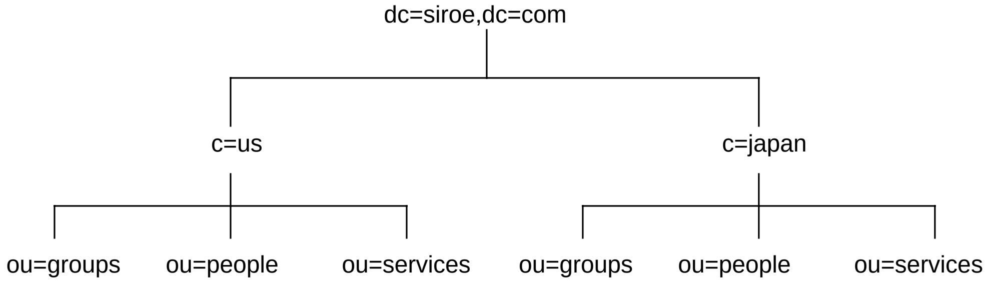
Directory Tree Design Examples

The following sections provide examples of directory trees designed to support a flat hierarchy as well as several examples of more complicated hierarchies.

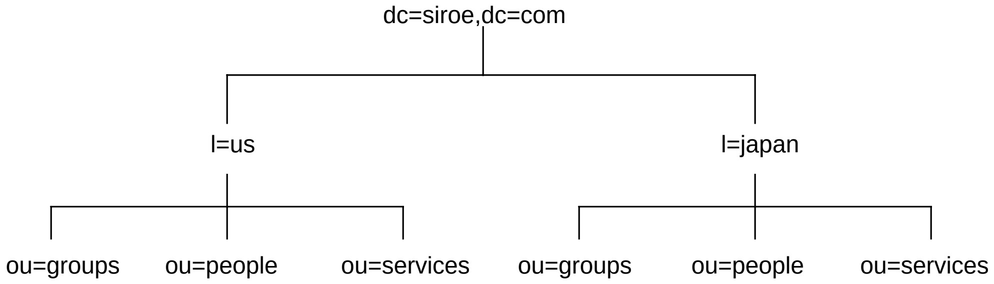
Directory Tree for an International Enterprise

To support an international enterprise, root your directory tree in your Internet domain name and then branch your tree for each country where your enterprise has operations immediately below that root point. In "Suffix Naming Conventions," on page 59, you are advised to avoid rooting your directory tree in a country designator. This is especially true if your enterprise is international in scope.

Because LDAP places no restrictions on the order of the attributes in your DNs, you can use the `c (countryName)` attribute to represent each country branch as follows:



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



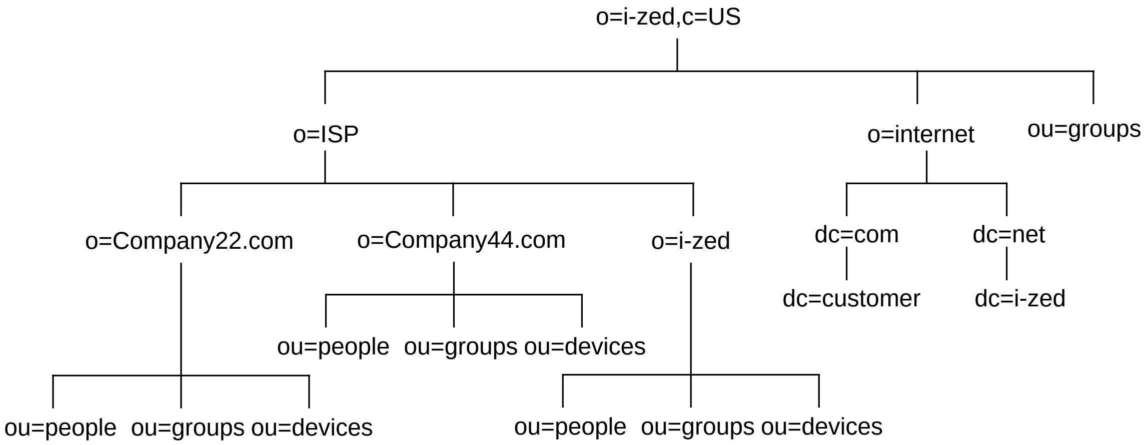
Directory Tree for an ISP

Internet service providers (ISPs) may support multiple enterprises with their directories. If you are an ISP, consider each of your customers as a unique enterprise and design their directory trees accordingly. For security reasons, each account should be provided a unique directory tree with a unique suffix and an independent security policy.

You can assign each customer a separate database, and store these databases on separate servers. Placing each directory tree in its own database allows you to back up and restore data for each directory tree without affecting your other customers.

In addition, partitioning helps reduce performance problems caused by disk contention, and reduces the number of accounts potentially affected by a disk outage.

For example, the directory tree for I-Zed, an ISP, appears as follows:



Other Directory Tree Resources

Take a look at the following links for more information about designing your directory tree:

- RFC 2247: Using Domains in LDAP/X.500 Distinguished Names
  http://www.ietf.org/rfc/rfc2247.txt
- RFC 2253: LDAPv3, UTF-8 String Representation of Distinguished Names
  http://www.ietf.org/rfc/rfc2253.txt

Chapter 5

Designing the Directory Topology

In Chapter 4, “Designing the Directory Tree”, you designed how your directory stores entries. Because Directory Server can store a large quantity of entries, you may need to distribute your entries across more than one server. Your directory’s topology describes how you divide your directory tree among multiple physical Directory Servers and how these servers link with one another.

This chapter describes planning the topology of your directory. It contains the following sections:

- Topology Overview
- Distributing Your Data
- About Knowledge References
- Using Indexes to Improve Database Performance

Topology Overview

You can design your deployment of iPlanet Directory Server to support a distributed directory where the directory tree you designed in Chapter 4, "Designing the Directory Tree" is spread across multiple physical Directory Servers. The way you choose to divide your directory across servers helps you accomplish the following:

- Achieve the best possible performance for your directory-enabled applications
- Increase the availability of your directory
- Improve the management of your directory

The database is the basic unit you use for jobs such as replication, performing backups, and restoring data. You can carve a single directory into manageable chunks and assign them to separate databases. These databases can then be distributed among a number of servers, reducing the work load for each server. You can store more than one database on a single server. For example, one server might contain three different databases.

When you divide your directory tree across several databases, each database contains a portion of your directory tree, called a suffix. For example, you can use a database to store the entries in the `ou=people,dc=siroe,dc=com` suffix, or branch of your directory tree.

When you divide your directory among several servers, each server is responsible for only a part of the directory tree. The distributed directory works similarly to the Domain Name Service (DNS), which assigns each portion of the DNS namespace to a particular DNS server. Likewise, you can distribute your directory namespace across servers while maintaining a directory that, from a client's point of view, appears to be a single directory tree.

The iPlanet Directory Server also provides *knowledge references*, mechanisms for linking directory data stored in different databases. Directory Server includes two types of knowledge references, referrals and chaining.

The remainder of this chapter describes databases and knowledge references, explains the differences between the two types of knowledge references, and describes how you can design indexes to improve the performance of your databases.

Distributing Your Data

Distributing your data allows you to scale your directory across multiple servers without physically containing those directory entries on each server in your enterprise. A distributed directory can thus hold a much larger number of entries than would be possible with a single server.

In addition, you can configure your directory to hide the distributing details from the user. As far as users and applications are concerned, there is simply a single directory that answers their directory queries.

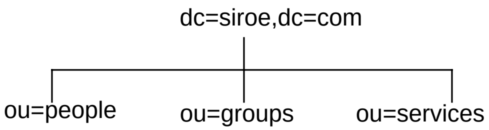
The following sections describe the mechanics of data distribution in more detail:

- "About Using Multiple Databases," on page 79
- "About Suffixes," on page 80

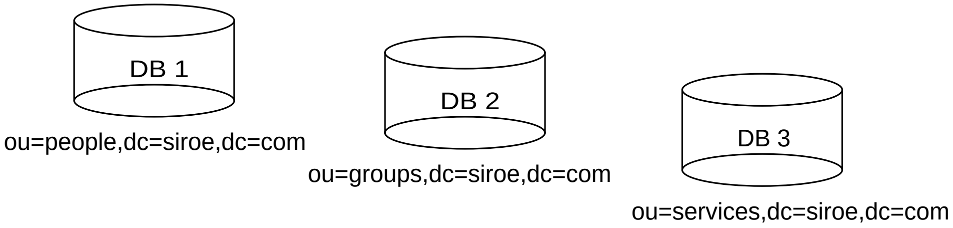
About Using Multiple Databases

iPlanet Directory Server stores data in LDBM databases.The LDBM database is a high-performance disk-based database. Each database consists of a set of large files that contains all of the data assigned to it.

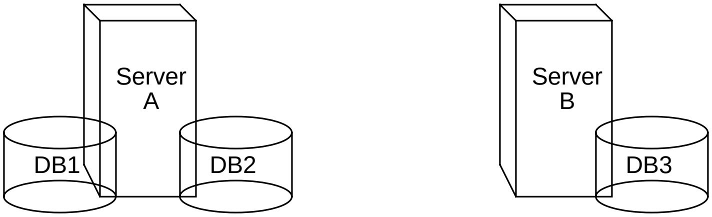
You can store different portions of your directory tree in different databases. For example, your directory tree appears as follows:



You can store the data of the three suffixes in three separate databases as follows:


DB 1
ou=people,dc=siroe,dc=com
DB 2
ou=groups,dc=siroe,dc=com
DB 3
ou=services,dc=siroe,dc=com

When you divide your directory tree among a number of databases, these databases can then be distributed across multiple servers. For example, the three databases you created to contain the three suffixes of your directory tree can be stored on two servers as follows:



Server A contains databases one and two and server B contains database three.

Distributing databases across multiple servers reduces the amount of work each server needs to do. Thus, the directory can be made to scale to a much larger number of entries than would be possible with a single server.

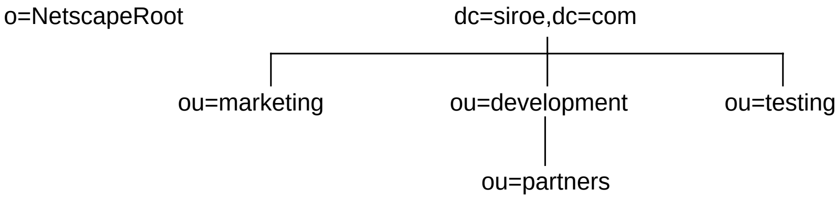
In addition, iPlanet Directory Server supports adding databases dynamically, meaning you can add new databases when your directory needs them without taking your entire directory off-line.

About Suffixes

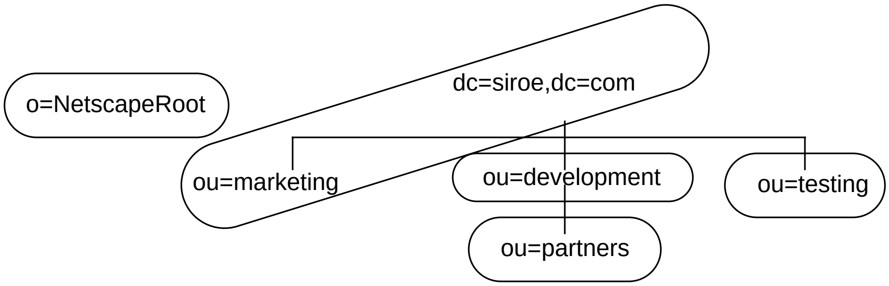
Each database contains the data within a suffix of your directory server. You can create both root and sub suffixes to organize the contents of your directory tree. A root suffix is the entry at the top of a tree. It can be the root of your directory tree or part of a larger tree you have designed for your directory server.

A sub suffix is a branch underneath a root suffix. The data for root and sub suffixes are contained by databases.

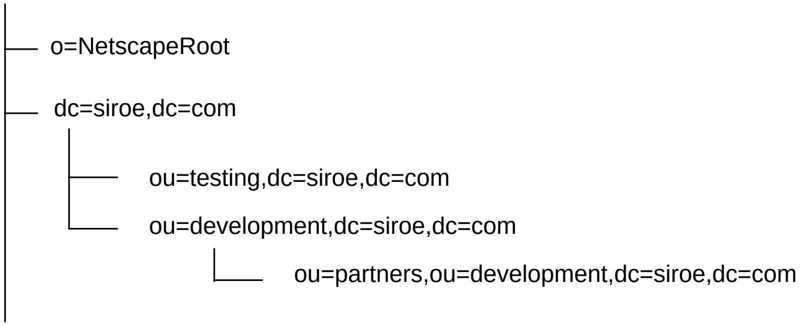
For example, you want to create suffixes to represent the distribution of your directory data. The directory tree for Siroe Corporation appears as follows:



Siroe Corporation decides to split their directory tree across five different databases as follows:

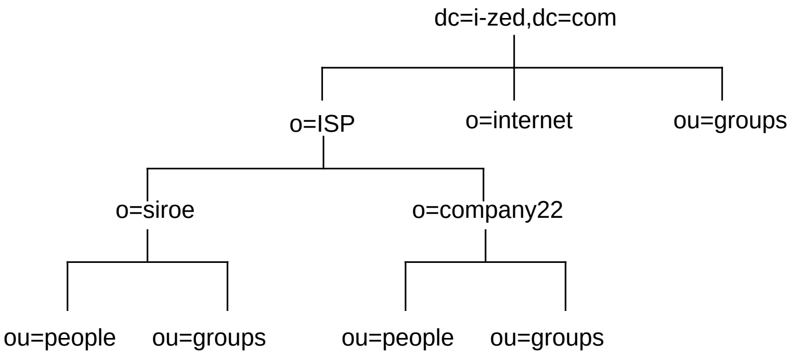


The suffixes that result contain entries for the following:


o=NetscapeRoot
dc=siroe,dc=com
ou=testing,dc=siroe,dc=com
ou=development,dc=siroe,dc=com
ou=partners,ou=development,dc=siroe,dc=com

The o=NetscapeRoot and dc=siroe,dc=com suffixes are both root suffixes. The other ou=testing,dc=siroe,dc=com suffix, the ou=development,dc=siroe,dc=com suffix and the ou=partners,ou=development,odc=siroe,dc=com suffix are all sub suffixes of the dc=siroe,dc=com root suffix. The root suffix dc=siroe,dc=com contains the data in the ou=marketing, branch of the original directory tree.

Your directory might contain more than one root suffix. For example, an ISP called I-Zed might host several websites, one for siroe.com and one for company22.com. The ISP would create two root suffixes, one corresponding to the dc=siroe,dc=com naming context and one corresponding to the dc=company22,dc=com naming context. The directory tree appears as follows:



The dc=i-zed,dc=com entry represents a root suffix. The entry for each hosted ISP is also a root suffix (o=siroe and o=company22). The ou=people and the ou=groups branches are sub suffixes under each root suffix.

About Knowledge References

Once you have distributed your data over several databases, you need to define the relationship between the distributed data. You do this using *knowledge references*, pointers to directory information held in different databases. The iPlanet Directory Server provides the following types of knowledge references to help you link your distributed data into a single directory tree:

- Referrals

The server returns a piece of information to the client application indicating that the client application needs to contact another server to fulfill the request.

- Chaining

  The server contacts other servers on behalf of the client application and returns the combined results to the client application after finishing the operation.

The following sections describe and compare these two types of knowledge references in more detail.

Using Referrals

A referral is a piece of information returned by a server that tells a client application the server to contact to proceed with an operation request. This redirection mechanism occurs when a client application requests a directory entry that does not exist on the local server.

Directory Server supports two types of referrals:

- A default referral

  The directory returns a default referral when a client application presents a DN for which the server does not have a matching suffix. Default referrals are stored in the configuration file of the server. You can set a default referral for the directory server and a separate default referral for each database.

  The default referral you set for each database is done through the suffix configuration information. When the suffix of the database is disabled, you can configure the directory to return a default referral to client requests made to that suffix. For more information about suffixes, refer to “About Suffixes,” on page 80. For information on configuring suffixes, refer to *iPlanet Directory Server Administrator’s Guide*.

- Smart referrals

  Smart referrals are stored on entries within the directory itself. Smart referrals point to Directory Servers that have knowledge of the subtree whose DN matches the DN of the entry containing the smart referral.

All referrals are returned in the format of an LDAP uniform resource locator (URL). The following sections describe the structure of an LDAP referral, and then describe the two referral types supported by Directory Server.

The Structure of an LDAP Referral

An LDAP referral contains information in the format of an LDAP URL. An LDAP URL contains the following information:

- The host name of the server to contact
- The port number of the server
- The base DN (for search operations) or target DN (for add, delete, and modify operations).

For example, a client application searches dc=siroe,dc=com for entries with a surname Jensen. A referral returns the following LDAP URL to the client application:

```
ldap://europe.siroe.com:389/ou=people,l=europe,dc=siroe,dc=com
```

The referral tells the client application to contact the host europe.siroe.com on port 389 and submit a search rooted at ou=people, l=europe,dc=siroe,dc=com.

The LDAP client application you use determines how a referral is handled. Some client applications automatically retry the operation on the server to which they have been referred. Other client applications simply return the referral information to the user. Most LDAP client applications provided by iPlanet (such as the command-line utilities or the Directory Server Gateway (DSGW)) automatically follow the referral. The same bind credentials you supply on the initial directory request are used to access the server.

Most client applications follow a limited number of referrals, or *hops*. The limit on the number of referrals followed reduces the time a client application spends trying to complete a directory lookup request and helps eliminate hung processes caused by circular referral patterns.

About Default Referrals

Default referrals are returned to clients when the server or database contacted does not contain the data requested.

The directory server determines whether a default 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 returns a default referral.

For example, a directory client requests the following directory entry:

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

However, the server manages only entries stored under the dc=europe,dc=siroe,dc=com suffix. The directory returns a referral to the client that indicates which server to contact for entries stored in the dc=siroe,dc=com suffix. The client then contacts the appropriate server and resubmits the original request.

You configure the default referral to point to a Directory Server that has more knowledge about the distribution of your directory. Default referrals for the server are set by the nsslapd-referral attribute. Default referrals for each database in your directory installation are set by the nsslapd-referral attribute in the database entry in the configuration. These attribute values are stored in the dse.ldif file.

For information on configuring default referrals, see the *iPlanet Directory Server Administrator's Guide*.

Smart Referrals

Directory Server also allows you to configure your directory to use smart referrals. *Smart referrals* allow you to associate a directory entry or directory tree to a specific LDAP URL. Associating directory entries to specific LDAP URLs allows you to refer requests to any of the following:

- Same namespace contained on a different server
- Different namespaces on a local server
- Different namespaces on the same server

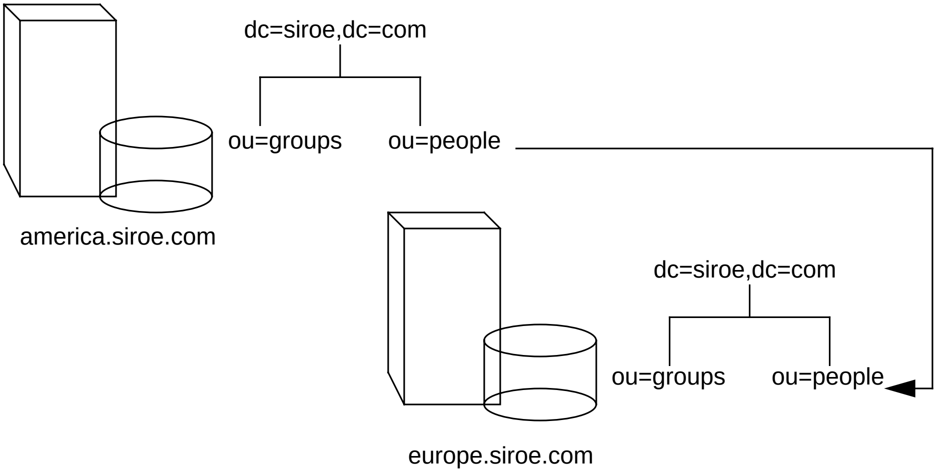
Unlike default referrals, the smart referrals are stored within the directory itself. For information on configuring and managing smart referrals, refer to the *iPlanet Directory Server Administrator's Guide*.

For example, the directory for the American office of Siroe Corporation contains the following directory branch point: ou=people,dc=siroe,dc=com.

You redirect all requests on this branch to the ou=people branch of the European office of Siroe Corporation by specifying a smart referral on the ou=people entry itself. This smart referral appears as follows:

```
ldap://europe.siroe.com:389/ou=people,dc=siroe,dc=com
```

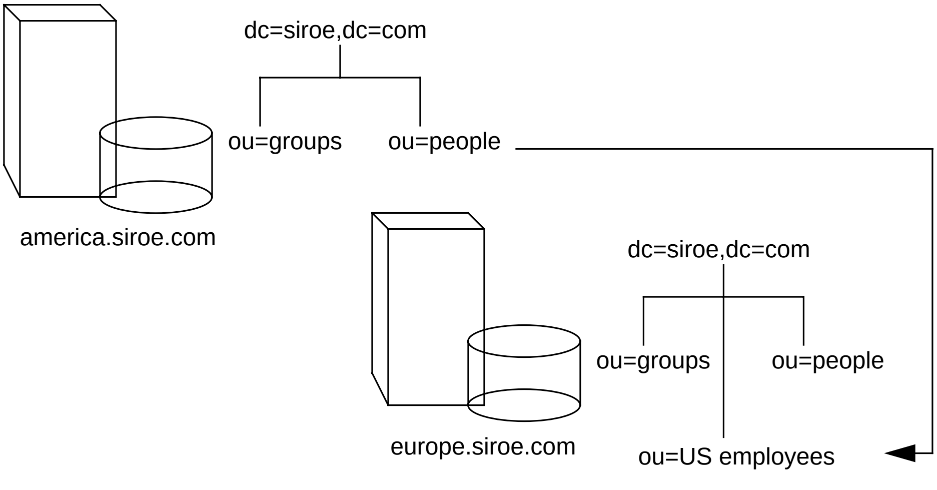
Any requests made to people branch of the American directory are redirected to the European directory. An illustration of this smart referral follows:



You can use the same mechanism to redirect queries to a different server that uses a different namespace. For example, an employee working in the Italian office of Siroe Corporation makes a request to the European directory for the phone number of a Siroe employee in America. The referral returned by the directory follows:

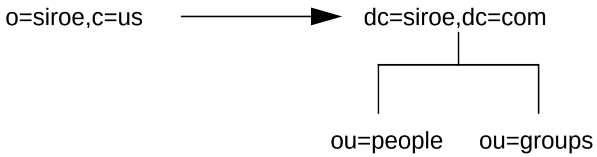
```
ldap://europe.siroe.com:389/ou=US employees,dc=siroe,dc=com
```

The following diagram illustrates how the referral works:



Finally, if you serve multiple suffixes on the same server, you can redirect queries from one namespace to another namespace served on the same machine. If you want to redirect all queries on the local machine for o=siroe,c=us to dc=siroe,dc=com, then you would put the following smart referral on the o=siroe,c=us entry:

```
ldap:///dc=siroe,dc=com
```



The third slash in this LDAP URL indicates that the URL points to the same Directory Server.

NOTE Creating a referral from one namespace to another works only for clients whose searches are based at that distinguished name. Other kinds of operations, such as searches below ou=people,o=siroe,c=US, will not be performed correctly.

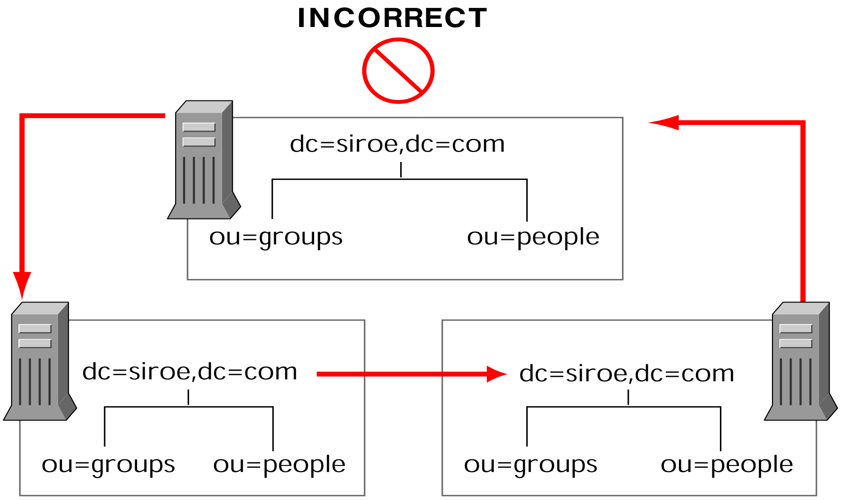
For more information on LDAP URLS and on how to include smart URLs on iPlanet Directory Server entries, see *iPlanet Directory Server Administrator's Guide*.

Tips for Designing Smart Referrals

While simple to implement, consider the following points before using smart referrals:

- Keep the design simple.

  Deploying your directory using a complex web of referrals makes administration difficult. Also, overusing smart referrals can lead to circular referral patterns. For example, a referral points to an LDAP URL, this LDAP URL in turn points to another LDAP URL, and so on until a referral somewhere in the chain points back to the original server. The following diagram depicts a circular referral pattern:



- Redirect at major branch points.

  Limit your referral usage to handle redirection at the suffix level of your directory tree. Smart referrals allow you to redirect lookup requests for leaf (non-branch) entries to different servers and DNs. As a result, you may be tempted to use smart referrals as an aliasing mechanism, leading to a complex and difficult to secure directory structure. 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, subsequently reducing your directory's administrative overhead.

- Consider the security implications.

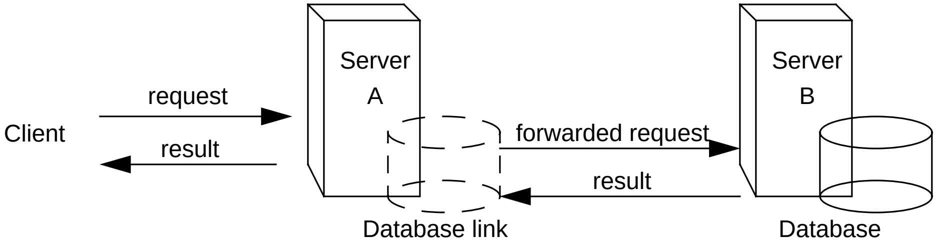
  Access control does not cross referral boundaries. Even if the server where the request originated allows access to an entry, when a smart referral sends a client request to another server, the client application may not be allowed access.

  Also, the client credentials need to be available on the server to which the client is referred for client authentication to take place.

Using Chaining

Chaining is a method for relaying requests to another server. This method is implemented through database links. A database link, as described in the section titled “Distributing Your Data,” on page 79, contains no data. Instead, it redirects client application requests to remote servers that contain the data.

During chaining, a server receives a request from a client application for data it does not contain. Using the database link, the server then contacts other servers on behalf of the client application and returns the results to the client application. This operation is illustrated in the following diagram.



Each database link is associated to a remote server holding data. You can also configure alternate remote servers containing replicas of the data for the database link to use when there is a failure. For more information on configuring database links, refer to the *iPlanet Directory Server Administrator’s Guide*.

Database links provide the following features:

- Invisible access to remote data.

  Because the database link takes care of client requests, data distribution is completely hidden from the client.

- Dynamic management.

  You can add or remove a part of the directory from the system while the entire system remains available to client applications. The database link can temporarily return referrals to the application until entries have been redistributed across the directory. You can also implement this functionality through the suffix itself, which can return a referral rather than forwarding a client application on to the database.

- Access control.

The database link impersonates the client application, providing the appropriate authorization identity to the remote server. You can disable user impersonation on the remote servers when access control evaluation is not required. For more information on configuring database links, refer to the *iPlanet Directory Server Administrator's Guide*.

Deciding Between Referrals and Chaining

Both methods of linking your directory partitions have advantages and disadvantages. The method, or combination of methods, you choose depends upon the specific needs of your directory.

The major difference between the two knowledge references is the location of the intelligence that knows how to locate the distributed information. In a chained system, the intelligence is implemented in the servers. In a system that uses referrals, the intelligence is implemented in the client application.

While chaining reduces client complexity, it does so at the cost of increased server complexity. Chained servers must work with remote servers and send the results to directory clients.

With referrals, the client must handle locating the referral and collating search results. However, referrals offer more flexibility for the writers of client applications and allow developers to provide better feedback to users about the progress of a distributed directory operation.

The following sections describe some of the more specific differences between referrals and chaining in greater detail.

Usage Differences

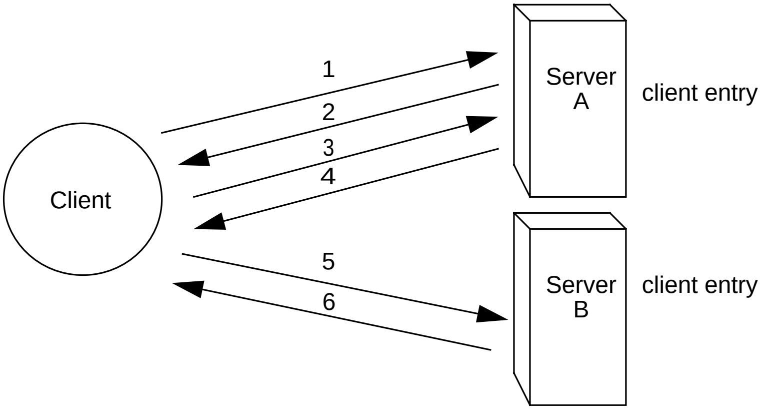
Some client applications do not support referrals. Chaining allows client applications to communicate with a single server and still access the data stored on many servers. Sometimes referrals do not work when a company's network uses proxies. For example, a client application has permissions to speak to only one server inside a firewall. If they are referred to a different server, they will not be able to contact it successfully.

Also, with referrals a client must authenticate, meaning that the servers to which clients are being referred need to contain the client credentials. With chaining, client authentication takes place only once. Clients do not need to authenticate again on the servers to which their requests are chained.

Evaluating Access Controls

Chaining evaluates access controls differently from referrals. With referrals, an entry for the client must exist on all of the target servers. With chaining, the client entry does not need to be on all of the target servers.

For example, a client sends a search request to server A. The following diagram illustrates the operation using referrals:

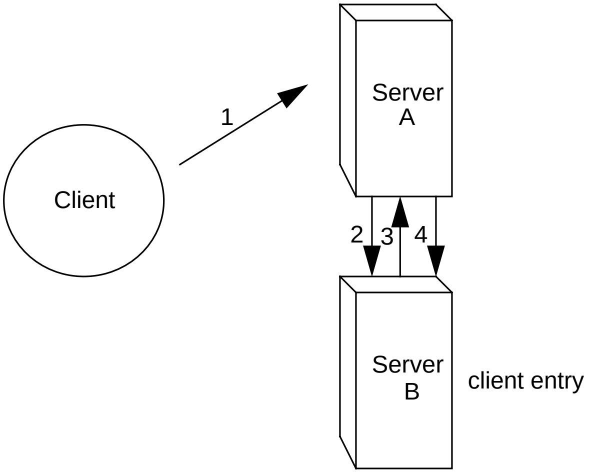


In the illustration above, the client application performs the following steps:

1. The client application first binds with Server A.
2. Server A contains an entry for the client that provides a user name and password, so returns a bind acceptance message. In order for the referral to work, the client entry must be present on Server A.
3. The client application sends the operation request to Server A.
4. However, Server A does not contain the information requested. Instead, Server A returns a referral to the client application telling them to contact Server B.
5. The client application then sends a bind request to Server B. To bind successfully, Server B must also contain an entry for the client application.
6. The bind is successful, and the client application can now resubmit its search operation to Server B.

This approach requires Server B to have a replicated copy of the client’s entry from Server A.

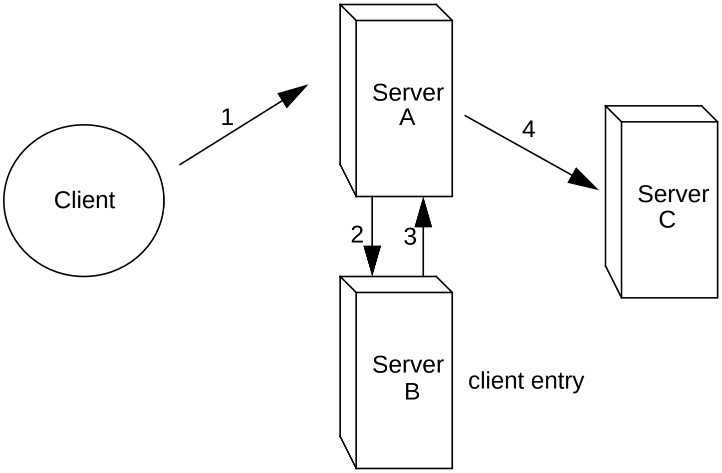
Chaining solves this problem. A search request would occur as follows on a chained system:



In the illustration above, the following steps are performed:

1. The client application binds with Server A and Server A tries to confirm that the user name and password are correct.
2. Server A does not contain an entry corresponding to the client application. Instead, it contains a database link to Server B, which contains the actual entry of the client. Server A sends a bind request to Server B.
3. Server B sends an acceptance response to Server A.
4. Server A then processes the client application's request using the database link. The database link contacts a remote data store located on Server B to process the search operation.

In a chained system, the entry corresponding to the client application does not need to be located on the same server as the data the client requests. For example, a system could be set up as follows:



In this illustration, the following steps are performed:

1. The client application binds with Server A and Server A tries to confirm that the user name and password are correct.
2. Server A does not contain an entry corresponding to the client application. Instead, it contains a database link to Server B, which contains the actual entry of the client. Server A sends a bind request to Server B.
3. Server B sends an acceptance response to Server A.
4. Server A then processes the client application's request using another database link. The database link contacts a remote data store located on Server C to process the search operation.

However, database links do not support the following access controls:

- Controls that must access the content of the user entry are not supported when the user entry is located on a different server. This includes access controls based on groups, filters, and roles.
- Controls based on client IP addresses or DNS domains may be denied. This is because the database link impersonates the client when it contacts remote servers. If the remote database contains IP-based access controls, it will evaluate them using the database link's domain rather than the original client domain.

Using Indexes to Improve Database Performance

Depending upon the size of your databases, searches performed by client applications can take a lot of time and resources. In order to improve search performance, you can use indexes.

Indexes are files stored in your directory databases. Separate index files are maintained for each database in your directory. Each file is named according to the attribute it indexes. The index file for a particular attribute can contain multiple types of indexes, allowing you to maintain several types of index for each attribute. For example, a file called `cn.db3` contains all of the indexes for the common name attribute.

Depending upon the types of applications using your directory, you will use different types of index. Different applications may frequently search for a particular attribute, or may search your directory in a different language, or may require data in a particular format.

This section contains the following topics:

- Overview of Directory Index Types
- Evaluating the Costs of Indexing

Overview of Directory Index Types

The directory supports the following types of index:

- Presence index.

  The presence index lists entries that possess a particular attribute, such as `uid`.

- Equality index.

  The equality index lists entries that contain a specific attribute value, such as `cn=Babs Jensen`.

- Approximate index.

  The approximate index allows approximate (or “sounds-like”) searches. For example, an entry contains the attribute value of `cn=Babs L. Jensen`. An approximate search would return this value for searches again `cn~=Babs Jensen`, `cn~=Babs`, and `cn~=Jensen`.

  Note that approximate indexes work only for English names written in ASCII characters.

- Substring index.

  The substring index allows searches against substrings within entries. For example, a search for `cn=*derson` would match common names containing this string (such as Bill Anderson, Norma Henderson, and Steve Sanderson).

- International index.

  The international index speeds searches for information in international directories. You configure the index to apply a matching rule by associating a locale (OID) with the attribute being indexed.

- Browsing Index

  The browsing, or virtual list view, index speeds up the display of entries in the Directory Server Console. You can create a browsing index on any branch in your directory tree to improve the display performance.

Evaluating the Costs of Indexing

Indexes improve search performance in your directory databases, but at the following costs:

- Indexes increase the time it takes to modify entries.

  The more indexes you maintain, the longer it takes the directory to update the database.

- Index files use disk space.

  The more attributes you index, the more files you create. And, if you create approximate and substring indexes for attributes that contain long strings, these files can grow rapidly.

- Index files use memory.

  To run more efficiently, the directory places as many index files in memory as possible. Index files use memory out of the pool available depending upon the database cache size. A large number of index files requires a larger database cache.

- Index files take time to create.

  Although index files will save time during searches, maintaining unnecessary indexes can waste time. Be certain to maintain only the files needed by the client applications using your directory.

Chapter 6

Designing the Replication Process

Replicating your directory contents increases the availability and performance of your directory. In Chapter 4 and Chapter 5, you made decisions about the design of your directory tree and your directory topology. This chapter addresses the physical and geographical location of your data, and specifically, how to use replication to ensure your data is available when and where you need it.

This chapter discusses uses for replication and offers advice on designing a replication strategy for your directory environment. It contains the following sections:

- Introduction to Replication
- Common Replication Scenarios
- Defining a Replication Strategy
- Using Replication with other Directory Features

Introduction to Replication

Replication is the mechanism that automatically copies directory data from one Directory Server to another. Using replication, you can copy any directory tree or subtree (stored in its own database) between servers. The Directory Server that holds the master copy of the information, will automatically copy any updates to all replicas.

Replication enables you to provide a highly available directory service, and to geographically distribute your data. In practical terms, replication brings the following benefits:

- Fault tolerance/Failover

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 client applications from accessing a particular Directory Server. Your clients are referred to another directory server for read and write operations. Note that to support write failover you must have a multi-master replication environment.

- Load balancing

  By replicating your directory tree across servers, you can reduce the access load on any given machine, thereby improving server response time.

- Higher performance and reduced response times

  By replicating directory entries to a location close to your users, you can vastly improve directory response times.

- Local data management

  Replication allows you to own and manage data locally while sharing it with other Directory Servers across your enterprise.

Before defining a replication strategy for your directory information, you should understand how replication works. This section describes:

- “Replication Concepts,” on page 98
- “Data Consistency,” on page 101

Replication Concepts

When you consider replication, you always start by making the following fundamental decisions:

- What information you want to replicate.
- Which server (or servers) hold the master copy, or read-write replica, of that information.
- Which server or servers hold the read-only copy, or read-only replica, of the information.
- What should happen when a read-only replica receives an update request, that is, which server should it refer the request to.

These decisions cannot be made effectively without an understanding of how the Directory Server handles these concepts. For example, when you decide what information you want to replicate, you need to know what is the smallest replication unit that the Directory Server can handle. The following sections contain definitions of concepts used by the Directory Server. This provides a framework for thinking about the global decisions you need to make.

Unit of Replication

In iPlanet Directory Server 5.0, the smallest unit of replication is a database. This means that you can replicate an entire database, but not a subtree within a database. Therefore, when you create your directory tree, you must take your replication plans into consideration. For more information on how to set up your directory tree, refer to Chapter 5, “Designing the Directory Topology.”

The replication mechanism also requires that one database correspond to one suffix. This means that you cannot replicate a suffix (or namespace) that is distributed over two or more databases.

Read-Write Replica/Read-Only Replica

A database that participates in replication is defined as a replica. There are two kinds of replicas: read-write or read-only. The read-write replicas contain master copies of directory information and can be updated. Read-only replicas refer all update operations to read-write replicas.

Supplier/Consumer

A server that holds a replica that is copied to a replica on a different server is called a *supplier* for that replica. A server that holds a replica that is copied from a different server is called a *consumer* for that replica. Generally, the replica on the supplier server is a read-write replica, and the one on the consumer server is a read-only replica. There are exceptions to this statement:

- In the case of cascading replication, the hub supplier holds a read-only replica that it supplies to consumers. For more information, refer to “Cascading Replication,” on page 105.
- In the case of multi-master replication, both masters are suppliers and consumers for the same read-write replica. For more information, refer to “Multi-Master Replication,” on page 103.

In iPlanet Directory Server 5.0, replication is always initiated by the supplier server, never by the consumer, in contrast to earlier versions of the Directory Server that allowed consumer-initiated replication (where you configure consumer servers to pull data from a supplier server).

For any particular replica, the supplier server must:

- Respond to read requests and update requests from directory clients.
- Maintain state information and a change log for the replica.
- Initiate replication to consumer servers.

  The supplier server is always responsible for recording the changes made to the read-write replicas that it manages. So, the supplier server makes sure that any changes are replicated to consumer servers.

A consumer server must:

- Respond to read requests.
- Refer update requests to a supplier server for the replica.

  Any time a request to add, delete, or change an entry is received by a consumer server, the request is referred to a supplier for the replica. The supplier server performs the request, then replicates the change.

In the special case of cascading replication, the hub supplier must:

- Respond to read requests.
- Refer update requests to a supplier server for the replica.
- Initiate replication to consumer servers.

For more information on cascading replication, refer to “Cascading Replication,” on page 105.

Change Log

Every supplier server maintains a change log. A change log is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on consumer servers, or on other masters in the case of multi-master replication.

When an entry is modified, a change record describing the LDAP operation that was performed is recorded in the change log.

Replication Agreement

Directory Servers use replication agreements to define replication. A replication agreement describes replication between *one* supplier and *one* consumer. The agreement is configured on the supplier server. It identifies:

- The database to replicate

- The consumer server to which the data is pushed
- The times that replication can occur
- The DN that the supplier server must use to bind (called the *supplier bind DN*)
- How the connection is secured (SSL, client authentication)

Data Consistency

Consistency refers to how closely the contents of replicated databases match each other at a given point in time. When you set up replication between two servers, part of the configuration is to schedule updates. With iPlanet Directory Server 5.0, it is always the supplier server that determines when consumer servers need to be updated, and initiates replication.

Directory Server offers the option of keeping replicas always synchronized, or of scheduling updates for a particular time of day, or day in the week. The advantage of keeping replicas always in sync is obviously that it provides better data consistency. The cost is the network traffic resulting from the frequent update operations. This solution is the best in cases where:

- You have a reliable high-speed connection between servers
- The client requests serviced by your directory are mainly search, read, and compare operations, with relatively few update operations.

In cases where you can afford to have looser consistency in data, you can choose the frequency of updates that best suits your needs or lowers the affect on network traffic. This solution is the best in cases where:

- You have unreliable or intermittently available network connections (such as a dial-up connection to synchronize replicas).
- The client requests serviced by your directory are mainly update operations.
- You need to reduce the communication costs.

In the case of multi-master replication, the replicas on each master are said to be *loosely consistent* because at any given time, there can be differences in the data stored on each master. This is true even when you have selected to always keep replicas in sync, because:

- There is a latency in the propagation of update operations between masters, and suppliers.

- The master that serviced the update operation does not wait for the second master to validate it before returning an “operation successful” message to the client.

Common Replication Scenarios

You need to decide how the updates flow from server to server and how the servers interact when propagating updates. There are three basic scenarios:

- Single-Master Replication
- Multi-Master Replication
- Cascading Replication
- Mixed Environments

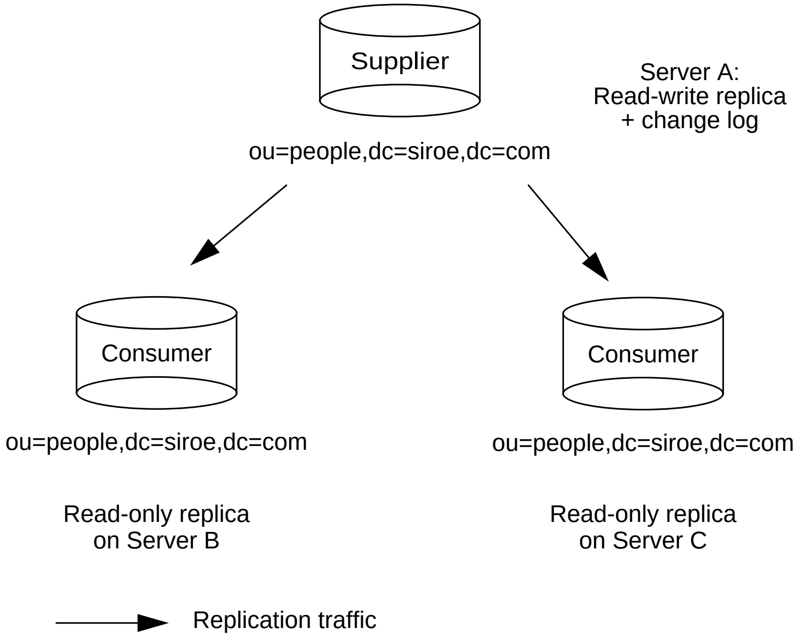
The following sections describe these methods and provide strategies for deciding the method is appropriate for your environment. You can also combine these basic scenarios to build the replication topology that best suits your needs.

Single-Master Replication

In the most basic replication configuration, a supplier server copies a replica directly to one or more consumer servers. In this configuration, all directory modifications occur on the read-write replica on the supplier server, and the consumer servers contain read-only replicas of the data.

The supplier server must perform all modifications to the read-write replicas stored on the consumer servers. Figure 6-1 on page 103 shows this simple configuration.

Figure 6-1 Single-Master Replication



The supplier server can replicate a read-write replica to several consumer servers. The total number of consumer servers that a single supplier server can manage depends on the speed of your networks and the total number of entries that are modified on a daily basis. However, you can reasonably expect a supplier server to maintain several consumer servers.

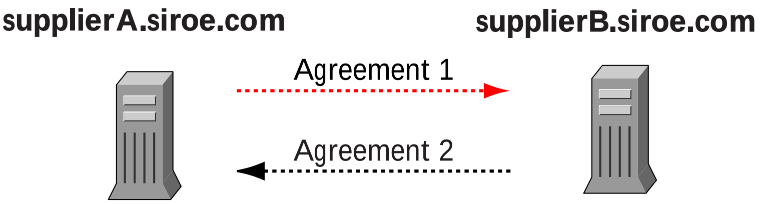
Multi-Master Replication

In a multi-master replication environment, master copies of the same information can exist on two servers. This means that data can be updated simultaneously in two different locations. The changes that occur on each server are replicated to the other. This means that each server plays both roles of supplier and consumer.

When the same data is modified on both servers, there is a conflict resolution procedure to determine which change is kept. The Directory Server considers the valid change to be the most recent one.

Although two separate servers can have master copies of the same data, within the scope of a single replication agreement, there is only one supplier server and one consumer. So, to create a multi-master environment between two supplier servers that share responsibility for the same data, you need to create more than one replication agreement.The following figure shows this configuration:

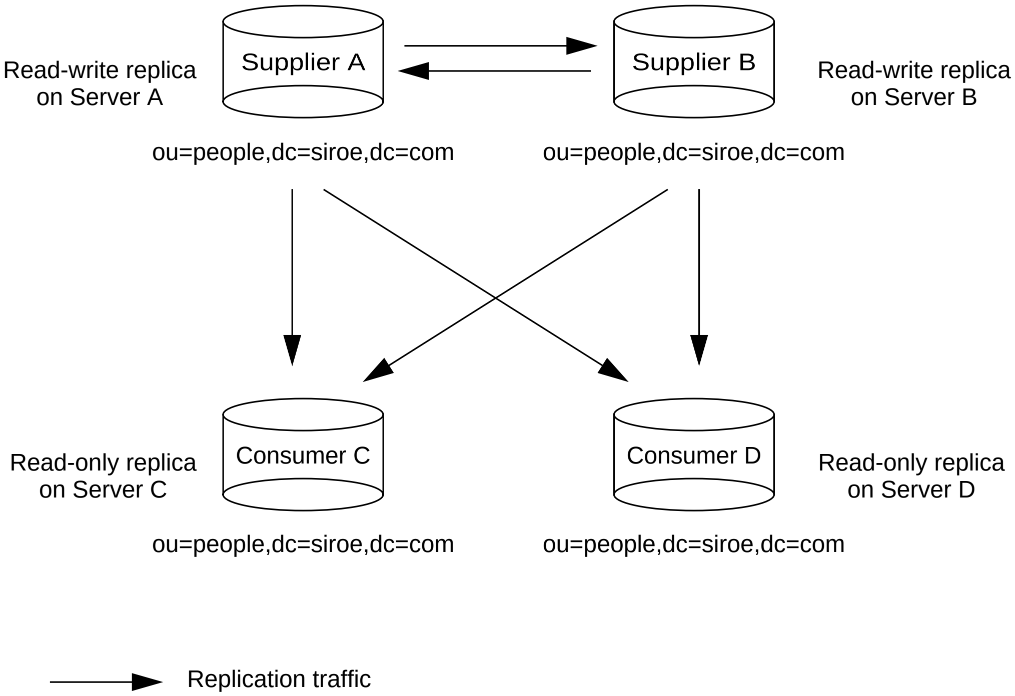
Figure 6-2 Multi-Master Replication Configuration (Two Masters)



In this illustration, Supplier A and Supplier B each hold a read-write replica of the same data.

The number of masters or suppliers you can have in any replication environment is limited to two. However, the number of consumer servers that hold read-only replicas is not limited. Figure 6-3 on page 105 shows the replication traffic in an environment with two masters (read-write replicas in the illustration), and two consumers (read-only replicas in the illustration). This figure shows that the consumers can be updated by both masters. The master servers ensure that the changes do not collide.

Figure 6-3 Replication Traffic in a Multi-Master Environment


Read-write replica on Server A
Supplier A
ou=people,dc=siroe,dc=com
Supplier B
Read-write replica on Server B
ou=people,dc=siroe,dc=com
Read-only replica on Server C
Consumer C
ou=people,dc=siroe,dc=com
Consumer D
Read-only replica on Server D
ou=people,dc=siroe,dc=com
Replication traffic

Cascading Replication

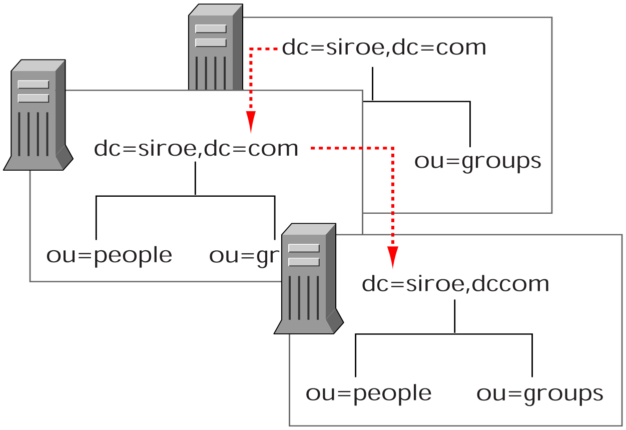
In a cascading replication scenario, a *hub supplier* receives updates from a supplier server, and replays those updates on consumer servers. The hub supplier is a hybrid: it holds a read-only replica, like a typical consumer server *and* it maintains a change log like a typical supplier server.

Hub suppliers do not actually keep copies of the master data themselves, they only pass it on as it received it from the original master. For the same reason, when a hub supplier receives an update request from a directory client, it refers the client to the master server.

Cascading replication is useful, for example, if some network connections between various locations in your organization are better than others. For example, suppose the master copy of your directory data is 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 cascaded replication to move directory data from Minneapolis to Saint Cloud to Duluth.

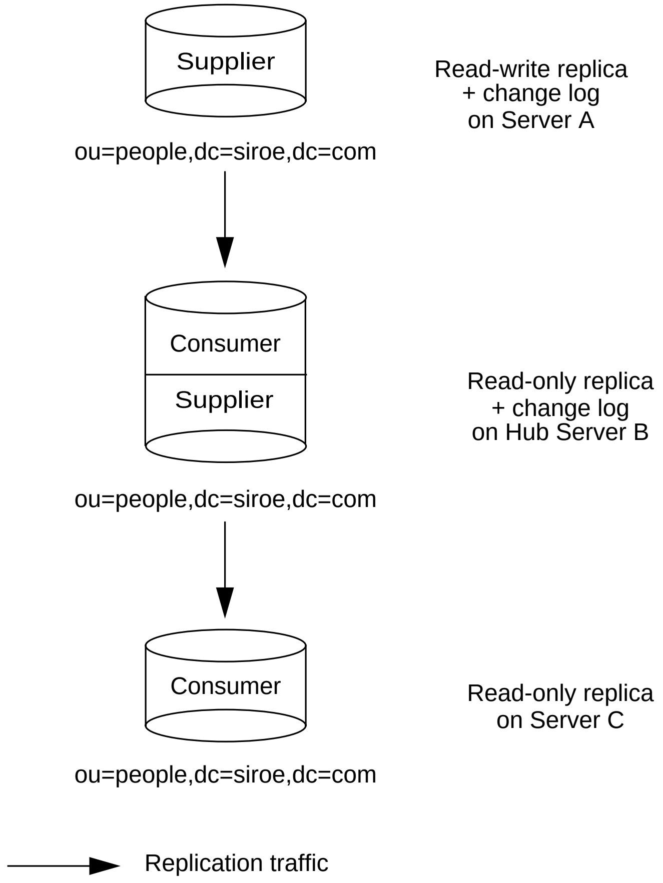
This cascading replication scenario is illustrated as follows:

Figure 6-4 Cascading Replication Scenario



The same scenario is illustrated below from a different perspective. It shows how the replicas are configured on each server (read-write or read-only) and which servers maintain a change log.

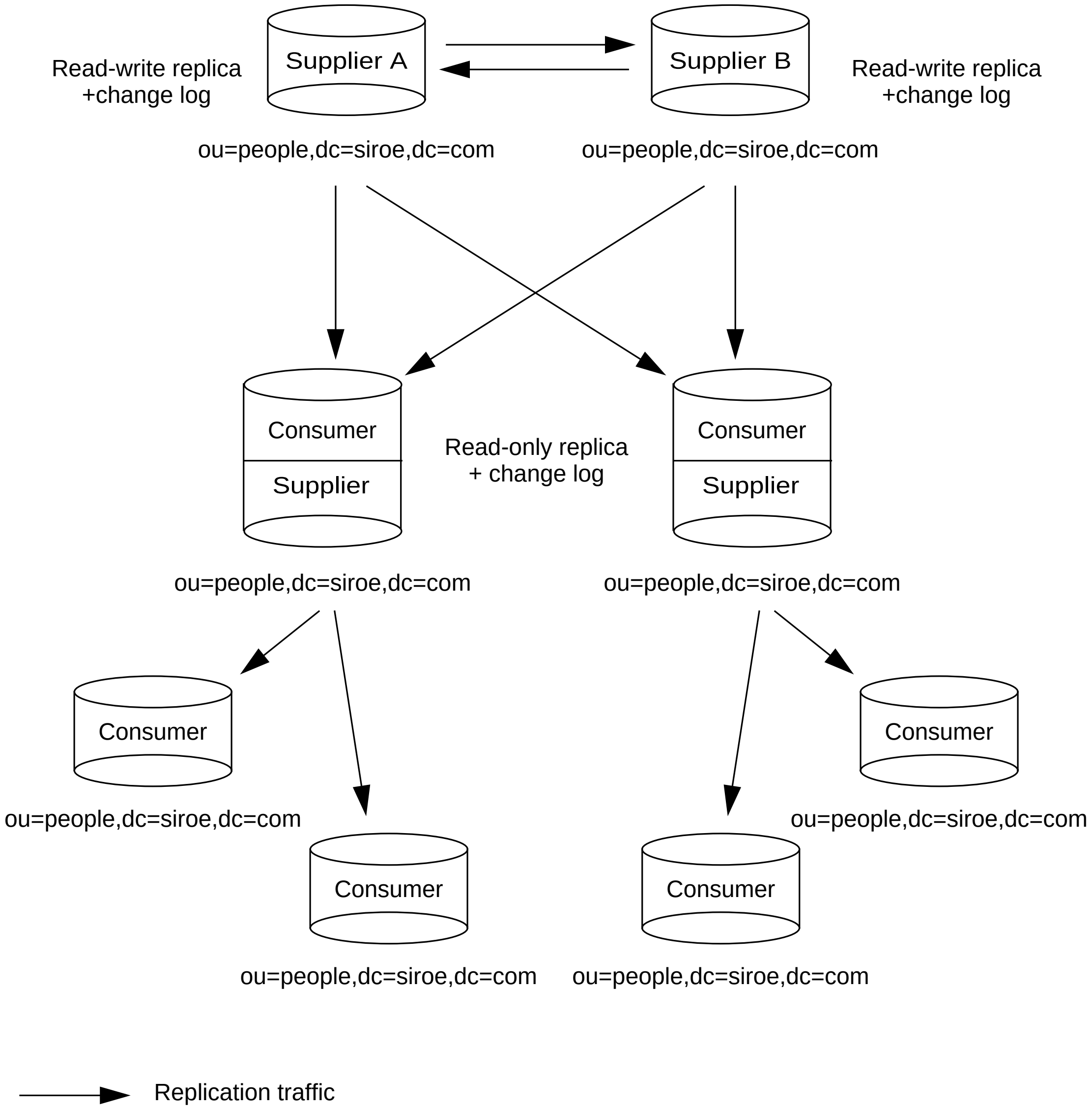
Figure 6-5 Replication Traffic and Change logs in Cascading Replication



Mixed Environments

You can combine any of the scenarios outlined in the previous sections to best fit your needs. For example, you could combine a multi-master configuration with a cascading configuration to produce something similar to the scenario illustrated in Figure 6-6 on page 108.

Figure 6-6 Combined Multi-Master and Cascading Replication


Read-write replica
+change log
Supplier A
Supplier B
Read-write replica
+change log
ou=people,dc=siroe,dc=com
ou=people,dc=siroe,dc=com
Consumer
Supplier
Read-only replica
+ change log
Consumer
Supplier
ou=people,dc=siroe,dc=com
ou=people,dc=siroe,dc=com
Consumer
ou=people,dc=siroe,dc=com
Consumer
ou=people,dc=siroe,dc=com
Consumer
ou=people,dc=siroe,dc=com
Consumer
ou=people,dc=siroe,dc=com
Replication traffic

Defining a Replication Strategy

The replication strategy that you define is determined by the service you want to provide:

- If high availability is your primary concern, you should create a data center with multiple directory servers on a single site. You can use single-master replication to provide read-failover, and multi-master replication to provide write-failover. How to configure replication for high availability is described in "Using Replication for High Availability," on page 111.
- If local availability is your primary concern, you should use replication to geographically distribute data to directory servers in local offices around the world. You can decide to hold a master copy of all information in a single location, such as the company headquarters, or to let local sites manage the parts of the DIT that are relevant for them. The type of replication configuration to set up is described in "Using Replication for Local Availability," on page 112.
- In all cases, you probably want to balance the load of requests serviced by your directory servers, and avoid network congestion. Strategies for load balancing your directory servers and your network are provided in "Using Replication for Load Balancing," on page 113.

To determine your replication strategy, start by performing a survey of your network, your users, your applications, and how they use the directory service you can provide. For guidelines on performing this survey, refer to the following section, "Replication Survey."

Once you understand your replication strategy, you can start deploying your directory. This is a case where deploying your service in stages will pay large dividends. By placing your directory into production in stages, you can get a better sense of the loads that your enterprise places on your directory. Unless you can base your load analysis on an already operating directory, be prepared to alter your directory as you develop a better understanding on how your directory is used.

The following sections describe in more detail the factors affecting your replication strategy:

- "Replication Survey," on page 110
- "Replication Resource Requirements," on page 110
- "Using Replication for High Availability," on page 111
- "Using Replication for Local Availability," on page 112

- "Using Replication for Load Balancing," on page 113
- "Example Replication Strategy for a Small Site," on page 116
- "Example Replication Strategy for a Large Site," on page 117

Replication Survey

The type of information you need to gather from your survey to help you define your replication strategy includes:

- Quality of the LANs and WANs connecting different buildings or remote sites, and the amount of available bandwidth.
- Physical location of users, how many users are at each site, what is their activity

  For example, a site that manages human resource databases or financial information is likely to put a heavier load on your directory than a site containing engineering staff that uses the directory for simple telephone book purposes.
- The number of applications that access the directory, and relative percentage of read/search/compare operations to write operations.

  For example, if your messaging server uses the directory, you need to know how many operations it performs for each email message it handles. Other products that rely on directory are typically products such as authentication applications, or meta-directory applications. For each one you must find out the type and frequency of operations that are performed in the directory.
- The number and size of the entries stored in the directory.

For example, a site that manages human resource databases or financial information is likely to put a heavier load on your directory than a site containing engineering staff that uses the directory for simple telephone book purposes.

Replication Resource Requirements

Using replication requires more resources. Consider the following resource requirements when defining your replication strategy:

- Disk usage.

On supplier servers, the change log is written after each update operation. Supplier servers receiving many update operations may see higher disk usage.

In addition, as there is a single change log on every supplier server, if a supplier contains multiple replicated databases the change log will be used more frequently, and the disk usage will be even higher.

- Server threads.

  Each replication agreement consumes one server thread. So, the number of threads available to client applications is reduced, possibly affecting the server performance for the client applications.

- File descriptors.

  The number of file descriptors available to the server is reduced by the change log (one file descriptor) and each replication agreement (one file descriptor per agreement).

Using Replication for High Availability

Use replication to prevent the loss of a single server from causing your directory to become unavailable. At a minimum you should replicate the local directory tree to at least one backup server.

Some directory architects argue that you should replicate three times per physical location for maximum data reliability. How much you use replication for fault tolerance is up to you, but you should base this decision on the quality of the hardware and networks used by your directory. Unreliable hardware needs more backup servers.

NOTE You should not use replication as a replacement for a regular data backup policy. For information on backing up your directory data, refer to the *iPlanet Directory Server Administrator's Guide*.

If you need to guarantee write-failover for all you directory clients, you should use a multi-master replication scenario. If read-failover is sufficient, you can use single-master replication.

LDAP client applications can usually be configured to search only one LDAP server. That is, unless you have written a custom client application to rotate through LDAP servers located at different DNS hostnames, you can only configure your LDAP client application to look at a single DNS hostname for a Directory

Server. Therefore, you will probably need to use either DNS round robins or network sorts to provide fail-over to your backup Directory Servers.For information on setting up and using DNS round robins or network sorts, see your DNS documentation.

Alternatively, you can use the iPlanet Directory Access Router product. For more information on iPlanet Directory Access Router, go to http://www.iplanet.com.

Using Replication for Local Availability

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.

You should use replication for local availability for the following reasons:

- You need a local master copy of the data.

  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. Having a local master copy of the data is also important to any enterprise where interoffice politics dictate that data be controlled at a divisional or organizational level.

- You are using unreliable or intermittently available network connections.

  Intermittent network connections 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 to be severely reduced.

  For example, enterprises with aging networks may experience these conditions during normal business hours.

Using Replication for Load Balancing

Replication can balance the load on your Directory Servers in several ways:

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

One of the more important reasons to replicate directory data is to balance the work load of 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.

Directory entries generally average around one KB in size. Therefore, every directory lookup adds about one KB to your network load. If your directory users perform around ten 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 because of 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 have only a few users at the remote site performing a few directory lookups per day.

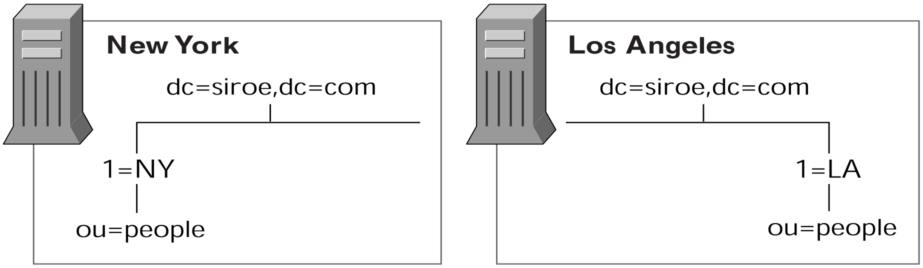
For example, consider that your directory tree on average includes in excess of 1,000,000 entries and that it is not unusual for about ten percent of those entries to change every day. If your average directory entry is only one KB in size, this means you could be increasing your network load by 100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

A good compromise between making data available to local sites without overloading the network is to use scheduled replication. For more information on data consistency and replication schedules, refer to "Data Consistency," on page 101.

Example of Network Load Balancing

Suppose your enterprise has offices in two cities. Each office has specific subtrees that they manage as follows:

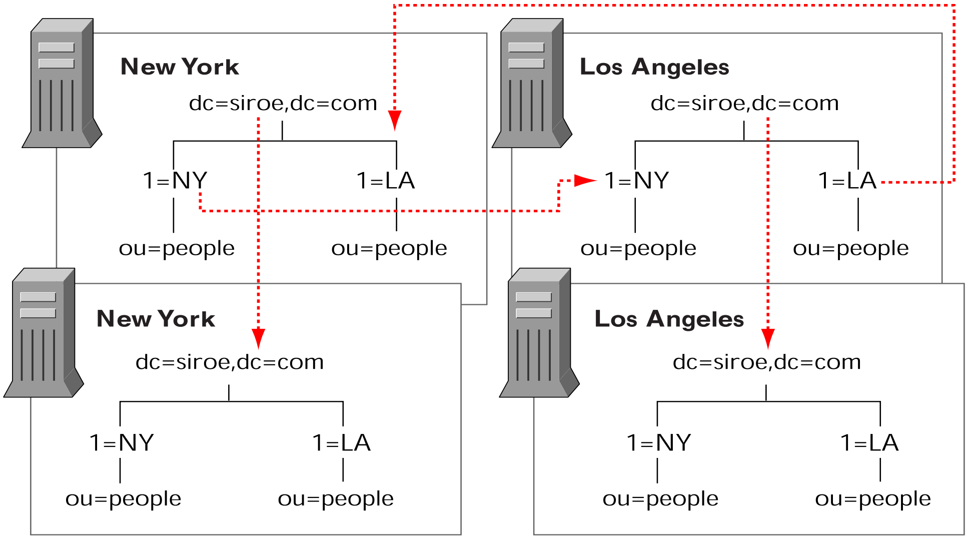


Each office contains a high-speed network, but you are using a dial-up connection to network between the two cities. To balance your network load:

- 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.

- Replicate the directory tree on each master server (including data supplied from the remote office) to at least one local Directory Server to ensure availability of the directory data. You can use multi-master replication for the suffix managed locally, and cascading replication for the suffix that is receives a master copy of the data from a remote server.



Example of Load Balancing for Improved Performance

Suppose that your directory must include 1,500,000 entries in support of 1,000,000 users, and each user performs ten directory lookups a day. Also assume that you are using a messaging server that handles 25,000,000 mail messages a day, and that performs five directory lookups for every mail message that it handles. Therefore, you can expect 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 eight-hour business day, and that your 1,000,000 directory users are clustered in four time zones, your business day (or peak usage) across four 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 six or seven Directory Servers to support this load. However, for enterprises with 1,000,000 directory users, you should add more Directory Servers for local availability purposes.

You could, therefore, replicate as follows:

- Place two Directory Servers in a multi-master configuration in one city to handle all write traffic.

  This configuration assumes that you want a single point of control for all directory data.

- Use these supplier servers to replicate to one or more hub suppliers.

  The read, search and compare requests serviced by your directory should be targeted at the consumer servers, thereby freeing the master servers to handle write requests. For a definition of a hub supplier, refer to “Cascading Replication,” on page 105.

- Use the hub supplier to replicate to local sites throughout the enterprise.

  Replicating to local sites helps balance the work load of 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 each hub supplier.

- At each site, replicate at least once to ensure high availability, at least for read operations.

  Use DNS sort to ensure that local users always find a local Directory Server they can use for directory lookups.

Example Replication Strategy for a Small Site

Suppose your entire enterprise is contained within a single building. This building has a very fast (100 MB per 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. Alternatively, use an LDAP proxy such as iPlanet Directory Access Router. For more information on iPlanet Directory Access Router, go to http://www.iplanet.com.

Example Replication Strategy for a Large Site

Suppose your entire enterprise is contained within two buildings. Each building has a very fast (100 MB per 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.

Your replication strategy follows:

- Choose a single server in one of the two buildings to contain a master copy of your directory data.

  This server should be placed in the building that contains the largest number of people responsible for the master copy of the directory data. Call this Building A.

- Replicate at least once within Building A for high availability of directory data.

  Use a multi-master replication configuration if you need to ensure write-failover.

- Create two replicas in the second building (Building B).
- If there is no need for close consistency between the supplier and consumer server, schedule replication so that it occurs only during off peak hours.

Using Replication with other Directory Features

Replication interacts with other iPlanet Directory Server features to provide advanced replication features. The following sections describe feature interactions to help you better design your replication strategy.

Replication and Access Control

The directory stores ACIs as attributes of entries. This means that the ACI is replicated along with other directory content. This is important because Directory Server evaluates ACIs locally.

For more information about designing access control for your directory, refer to Chapter 7, "Designing a Secure Directory," on page 121.

Replication and Directory Server Plug-ins

You can use replication with most of the plug-ins delivered with iPlanet Directory Server. There are some exceptions and limitations in the case of multi-master replication with the following plug-ins:

- Attribute uniqueness plug-in
- Referential integrity plug-in

You cannot use multi-master replication with the attribute uniqueness plug-in at all, because this plug-in can validate only attribute values on the same server not on both servers in the multi-master set.

You can use the referential integrity plug-in with multi-master replication providing that this plug-in is enabled on just one master in the multi-master set. This ensures that referential integrity updates are made on just one of the master servers, and propagated to the other.

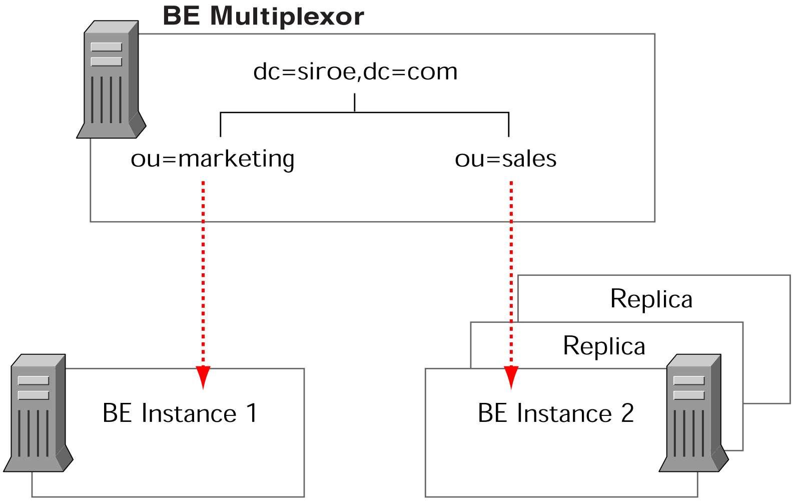
NOTE By default, these plug-ins are disabled. You need to use the Directory Server Console or the command line to enable them.

Replication and Database Links

When you distribute entries using chaining, the server containing the database link points to a remote server that contains the actual data. In this environment, you cannot replicate the database link itself. You can, however, replicate the database that contains the actual data on the remote server.

You must not use the replication process as a backup for database links. You must backup database links manually. For more information about chaining and entry distribution, refer to Chapter 5, "Designing the Directory Topology," on page 77.

Figure 6-7 Replicating Chained Databases



Schema Replication

In all replication scenarios, before pushing data to consumer servers, the supplier server checks whether its own version of the schema is in sync with the version of the schema held on consumer servers.

If the schema entries on both supplier and consumers are the same, the replication operation proceeds.

If the version of the schema on the supplier server is more recent than the version stored on the consumer, the supplier server replicates its schema to the consumer before proceeding with the data replication.

If the version of the schema on the supplier server is older than the version stored on the consumer, you will probably witness a lot of errors during replication because the schema on the consumer cannot support the new data.

A consumer might contain replicated data from two suppliers, each with different schema. Whichever supplier was updated last will “win” and its schema will be propagated to the consumer.

NOTE You must *never* update the schema on a consumer server because the supplier server is unable to resolve the conflicts that will occur and replication will fail.

Schema should be maintained on a master supplier server in a replicated topology. If using the standard 99user.ldif file, these changes will be replicated to all consumers. When using custom schema files, ensure that these files are copied to all servers after making changes on the master supplier. After copying files, the server must be restarted. Refer to “Creating Custom Schema Files,” on page 51 for more information.

In iPlanet Directory Server 5.0, the same server can hold read-write replicas for which it acts as a supplier, and read-only replicas for which it acts as a consumer. Therefore, you should always identify the server that will act as a supplier for the schema.

NOTE Special replication agreements are not required to replicate the schema. If replication has been configured between a supplier and a consumer, schema replication will happen by default.

Changes made to custom schema files are only replicated if the schema is updated using LDAP or the Directory Server Console. These custom schema files should be copied to each server in order to maintain the information in the same schema file on all servers. For more information, refer to “Creating Custom Schema Files,” on page 51.

For more information on schema design, refer to Chapter 3, “How to Design the Schema.”

Chapter 7

Designing a Secure Directory

How you secure the data in Directory Server affects all of the previous design areas. Your security design needs to protect the data contained by your directory and meet the security and privacy needs of your users and applications.

This chapter describes how to analyze your security needs and explains how to design your directory to meet these needs. It includes the following sections:

- About Security Threats
- Analyzing Your Security Needs
- Overview of Security Methods
- Selecting Appropriate Authentication Methods
- Preventing Authentication by Account Inactivation
- Designing a Password Policy
- Designing Access Control
- Securing Connections With SSL
- Other Security Resources

About Security Threats

There are many potential threats to the security of your directory. Understanding the most common threats helps you plan your overall security design. The most typical threats to directory security fall into the following three categories:

- Unauthorized Access
- Unauthorized Tampering

- Denial of Service

The remainder of this section provides a brief overview of the most common security threats to assist you with designing your directory's security policies.

Unauthorized Access

While it may seem simple to protect your directory from unauthorized access, the problem can in fact be more complicated. There are several opportunities along the path of directory information delivery for an unauthorized client to gain access to data.

For example, an unauthorized client can use another client's credentials to access the data. This is particularly likely when your directory uses unprotected passwords. Or an unauthorized client can eavesdrop on the information exchanged between a legitimate client and Directory Server.

Unauthorized access can occur from inside your company, or if your company is connected to an extranet or to the Internet, from outside.

The scenarios described here are just a few examples of how an unauthorized client might access your directory data.

The authentication methods, password policies, and access control mechanisms provided by the iPlanet Directory Server offer efficient ways of preventing unauthorized access. Refer to "Selecting Appropriate Authentication Methods," on page 127, "Designing a Password Policy," on page 131, and "Designing Access Control," on page 136, for more information about these topics.

Unauthorized Tampering

If intruders gain access to your directory or intercept communications between Directory Server and a client application, they have the potential to modify (or tamper with) your directory data. Your directory is rendered useless if the data can no longer be trusted by clients, or if the directory itself cannot trust the modifications and queries it receives from clients.

For example, if your directory cannot detect tampering, an attacker could change a client's request to the server (or not forward it) and change the server's response to the client. SSL and similar technologies can solve this problem by signing information at either end of the connection. For more information about using SSL with iPlanet Directory Server, refer to "Securing Connections With SSL," on page 144.

Denial of Service

With a denial of service attack, the attacker's goal is to prevent the directory from providing service to its clients. For example, an attacker might simply use the system's resources to prevent them from being used by someone else.

iPlanet Directory Server offers a way of preventing denial of service attacks by setting limits on the resources allocated to a particular bind DN. For more information about setting resource limits based on the user's bind DN, refer to "User Account Management" in the *iPlanet Directory Server Administrator's Guide*.

Analyzing Your Security Needs

You need to analyze your environment and users to determine your specific security needs. When you performed your site survey in Chapter 3, "Directory Data Design," you 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 design.

The way you implement security is also dependent on how you use the directory to support your business. A directory that serves an intranet does not require the same security measures as a directory that supports an extranet, or e-commerce applications that are open to the Internet.

If your directory serves an intranet only, your concerns are:

- To provide users and applications with access to the information they need to perform their jobs
- To protect sensitive data regarding employees or your business from general access

If your directory serves an extranet, or supports e-commerce applications over the Internet, in addition to the previous points, your concerns are:

- To offer your customers a guarantee of privacy
- To guarantee information integrity

This section contains the following information about analyzing your security needs:

- "Determining Access Rights," on page 124
- "Ensuring Data Privacy and Integrity," on page 124

- "Conducting Regular Audits," on page 125
- "Example Security Needs Analysis," on page 125

Determining Access Rights

When you perform your data analysis, you decide what information your users, groups, partners, customers, and applications need to access.

You can take grant access rights in two ways:

- Grant all categories of users as many rights as possible while still protecting your sensitive data.

  If you choose this open method, you must concentrate on determining what data is sensitive or critical to your business

- Grant each category of users the minimum access they require to do their jobs.

  If you choose this restrictive method, you must spend some time understanding the information needs of each category of user inside, and possibly outside of your organization.

No matter how you determine to grant access rights, you should create a simple table that lists the categories of users in your organization and the access rights you grant to each. You may also want to create a table that lists the sensitive data held in the directory, and for each piece of data, the steps taken to protect it.

For information about checking the identity of users, refer to "Selecting Appropriate Authentication Methods," on page 127. For information about restricting access to directory information, refer to "Designing Access Control," on page 136.

Ensuring Data Privacy and Integrity

When you are using the directory to support exchanges with business partners over an extranet, or to support e-commerce applications with customers on the Internet, you must ensure the privacy and the integrity of the data exchanged.

You can do this in several ways:

- By encrypting data transfers
- By using certificates to sign data transfers

For information about encryption methods provided in the iPlanet Directory Server, refer to "Password Storage Scheme," on page 134. For information about signing data, refer to "Securing Connections With SSL," on page 144.

Conducting Regular Audits

As an extra security measure, you should conduct regular audits to verify the efficiency of your overall security policy. You can do this by examining the log files and the information recorded by the SNMP agents.

For more information about SNMP, refer to *iPlanet Directory Server Administrator's Guide*.

Example Security Needs Analysis

The examples provided in this section illustrate how the imaginary ISP company Siroe analyzes its security needs.

Siroe's business is to offer web hosting and internet access. Part of Siroe's activity is to host the directories of client companies. It also provides internet access to a number of individual subscribers.

Therefore, Siroe has three main categories of information in its directory:

- Siroe internal information
- Information belonging to corporate customers
- Information pertaining to individual subscribers

Siroe needs the following access controls:

- Provide access to the directory administrators of Company22 and Company33 to their own directory information.
- Implement Company22's and Company33's own access control policies for their directory information.
- Implement a standard access control policy for all individual clients who use Siroe for internet access from their homes.
- Deny access to Siroe's corporate directory to all outsiders.
- Grant read access to Siroe's directory of subscribers to the world.

Overview of Security Methods

iPlanet Directory Server offers a number of methods that you can use to design an overall security policy that is adapted to your needs. Your security policy should be strong enough to prevent sensitive information from being modified or retrieved by unauthorized users while simple enough to administer easily. 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.

iPlanet Directory Server provides the following security methods:

- Authentication

  A means for one party verifies another's identity. For example, a client gives a password to Directory Server during an LDAP bind operation.

- Password policies

  Defines the criteria that a password must satisfy to be considered valid, for example, age, length, and syntax.

- Encryption

  Protects the privacy of information. When data is encrypted, it is scrambled in a way that only the recipient can understand.

- Access control

  Tailors the access rights granted to different directory users, and provides a means of specifying required credentials or bind attributes.

- Account inactivation

  Disables a user account, group of accounts or an entire domain so that all authentication attempts are automatically rejected.

- Signing with SSL

  Maintains the integrity of information. If information is signed, the recipient can determine that it was not tampered with during transit.

- Auditing

  Allows you to determine if the security of your directory has been compromised. For example, you can audit the log files maintained by your directory.

These tools for maintaining security can be used in combination in your security design. You can also use other features of the directory such as replication and data distribution to support your security design.

Selecting Appropriate Authentication Methods

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

iPlanet Directory Server provides the following methods for authentication:

- Anonymous Access
- Simple Password
- Certificate-Based Authentication
- Simple Password Over TLS
- Proxy Authentication

The directory uses the same authentication mechanism for all users, whether they are people or LDAP-aware applications.

For information about preventing authentication by a client or group of clients, see “Preventing Authentication by Account Inactivation,” on page 130.

Anonymous Access

Anonymous access provides the easiest form of access to your directory. It makes data available to any user of your directory, whether they have authenticated or not.

However, anonymous access does not allow you to track who is performing what kinds of searches; only that someone is performing searches. When you allow anonymous access, anyone who connects 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.

You can restrict the privileges of anonymous access. Usually directory administrators only allow anonymous access for read, search, and compare privileges (not for write, add, delete, or selfwrite). Often, administrators limit access to a subset of attributes that contain general information such as names, telephone numbers, and email addresses. 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, and salary information.

If a user attempts to bind with an entry that does not contain a user password attribute, Directory Server either:

- Grant anonymous access if the user does not attempt to provide a password
- Deny access if the user provides any non-null string for the password

For example, consider the following ldapsearch command:

```
% ldapsearch -D “cn=joe” -w secretpwd -b “siroe.com” cn=joe
```

Although the directory allows anonymous access for read, Joe cannot access his own entry because it does not contain a password that matches the one he provided in the `ldapsearch` command.

Simple Password

If you have not set up anonymous access, you must authenticate to the directory before you can access the directory contents. With simple password authentication, a client authenticates to the server by sending a simple, reusable password.

For example, a client authenticates to the directory via a bind operation in which it provides a distinguished name and a set of credentials. The server locates the entry in the directory that corresponds to the client DN and checks whether the password given by the client matches the value stored with the entry. If it does, the server authenticates the client. If it does not, the authentication operation fails and the client receives an error message.

The bind DN often corresponds to the entry of a person. However, some directory administrators find it useful to bind as an organizational entry rather than as a person. The directory requires the entry used to bind to be of an object class that allows the `userPassword` attribute. This ensures that the directory recognizes the bind DN and password.

Most LDAP clients hide the bind DN from the user because users may find the long strings of DN characters hard to remember. 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 application searches the directory for that identifier and returns the associated distinguished name (such as `uid=fchen,ou=people,dc=siroe,dc=com`).
3. The LDAP client application binds to the directory using the retrieved distinguished name and the password supplied by the user.

NOTE The drawback of simple password authentication is that the password is sent in clear text over the wire. If a rogue user is listening, this can compromise the security of your directory because that person can impersonate an authorized user.

Simple password authentication offers an easy way of authenticating users, but it is best to restrict its use to your organization’s intranet. It does not offer the level of security required for transmissions between business partners over an extranet, or for transmissions with customers out on the Internet.

Certificate-Based Authentication

An alternate form of directory authentication involves using security certificates to bind to the directory. The directory prompts your users for a password when they first access it. However, rather than matching a password stored in the directory, the password opens the user’s certificate database.

If the user supplies the correct password, the directory client application obtains authentication information from the certificate database. The client application and the directory then use this information to identify the user by mapping the user’s certificate to a directory DN. The directory allows or denies access based on the directory DN identified during this authentication process.

For more information about certificates and SSL, see *Managing Servers with iPlanet Console*.

Simple Password Over TLS

When a secure connection is established between Directory Server and a client application using SSL or the Start TLS operation, the server can demand an extra level of authentication by requesting a password. In such cases, the password is not passed in clear over the wire.

For more information about SSL, refer to “Securing Connections With SSL,” on page 144. For information about the Start TLS operation, refer to the *iPlanet Directory Server Administrator’s Guide*.

Proxy Authentication

Proxy authentication is a special form of authentication because the user requesting access to the directory does not bind with its own DN but with a *proxy DN*.

The proxy DN is an entity that has appropriate rights to perform the operation requested by the user. When you grant proxy rights to a person or an application, you grant the right to specify any DN as a proxy DN, with the exception of the Directory Manager DN.

The proxy mechanism is very powerful. One of its main advantages is that you can enable an LDAP application to use a single thread with a single bind to service multiple users making requests against the Directory Server. Instead of having to bind and authenticate for each user, the client application binds to the Directory Server using a proxy DN.

The proxy DN is specified in the LDAP operation submitted by the client application. For example:

```
% ldapmodify -D “cn=joe” -w secretpwd -y
“cn=manager,dc=siroe,dc=com” -b “siroe.com” -f mods.ldif
```

This `ldapmodify` command gives a user named Joe the permissions of the manager entry (`cn=manager`) to apply the modifications in the `mods.ldif` file. Note that he does not need to provide the password for the manager.

Preventing Authentication by Account Inactivation

You can temporarily inactivate a user account or a set of accounts. Once inactivated, a user cannot bind to the directory, and the authentication operation fails.

Account inactivation is implemented through the operational attribute `nsAccountLock`. When an entry contains the `nsAccountLock` attribute with a value of `true`, the server rejects the bind.

You use the same procedures for inactivating users and roles. However, inactivating a role means that you inactivate all of the members of that role and not the role entry itself. For more information about roles, refer to “About Roles,” on page 71.

Designing a Password Policy

A password policy is a set of rules that govern how passwords are used in a given system. The password policy mechanism provided by Directory Server allows you to dictate such things as how short a password must be and whether users can reuse passwords. When users attempt to bind to the directory, the directory compares the password with the value in the password attribute of the user’s directory entry to make sure they match. Directory Server also uses the rules defined by the password policy to ensure that the password is valid before allowing the user to bind to the directory.

Password Policy Attributes

This section describes the attributes you set to create a password policy for your server. The attributes are described in the following sections:

- User-Defined Passwords
- Password Change After Reset
- Password Expiration
- Expiration Warning
- Password Syntax Checking
- Password Length
- Password Minimum Age
- Password History
- Password Storage Scheme

Password Change After Reset

The Directory Server password policy lets you decide whether users must change their passwords after the first login or after the password is reset by the administrator.

Often the initial passwords set by the administrator follow some sort of convention, such as the user's initials, user ID, or the company name. Once the convention is discovered, it is usually the first value tried by a hacker trying to break in. In this case, it is a good idea to require users to change their passwords after such a change. If you configure this option for your password policy, users are required to change their password even if user-defined passwords are disabled. (See "User-Defined Passwords," on page 132 for information.)

If you choose to not allow users from changing their own passwords, administrator assigned passwords should not follow any obvious convention and should be difficult to discover.

By default, users do not need to change their passwords after reset.

User-Defined Passwords

You can set up your password policy to either allow or not allow users from changing their own passwords. A good password is the key to a strong password policy. Good passwords do not use trivial words—that is, any word that can be found in a dictionary, names of pets or children, birthdays, user IDs, or any other information about the user that can be easily discovered (or stored in the directory itself).

Also, a good password should contain a combination of letters, numbers, and special characters. Often, however, users simply use passwords that are easy to remember. This is why some enterprises choose to set passwords for users that meet the criteria of a "good" password and not allow the users to change the passwords.

However, assigning passwords to users takes a substantial amount of an administrator's time. In addition, by providing passwords for users rather than letting them come up with passwords that are meaningful to them and therefore more easily remembered, you run the risk that the users will write their passwords down somewhere where they can be discovered.

By default, user-defined passwords are allowed.

Password Expiration

You can set your password policy so that users can use the same passwords indefinitely. Or, you can set your policy so that passwords expire after a given time. In general, the longer a password is in use, the more likely it is to be discovered. On the other hand, if passwords expire too often, users may have trouble remembering them and resort to writing their passwords down. A common policy is to have passwords expire every 30 to 90 days.

The server remembers the password expiration even if you turn the password expiration feature off. This means that if you turn the password expiration option back on, passwords are valid only for the duration you set before you last disabled the feature. For example, suppose you set up passwords to expire every 90 days and then decided to disable password expiration. When you decide to re-enable password expiration, the default password expiration duration is 90 days because that is what you had it set to before you disabled the feature.

By default, user passwords never expire.

Expiration Warning

If you choose to set your password policy so that user passwords expire after a given number of days, it is a good idea to send users a warning before their passwords expire. You can set your policy so that users are sent a warning 1 to 24,855 days before their passwords expire. The Directory Server displays the warning when the user binds to the server. If password expiration is turned on, by default, a warning is sent (via LDAP message) to the user one day before the user's password expires, provided the user's client application supports this feature.

Password Syntax Checking

The password policy establishes some syntax guidelines for password strings, such as the minimum password length guideline. The password syntax-checking mechanism ensures that the password strings conform to the password syntax guidelines established by the password policy. Also, the password syntax-checking mechanism also ensures that the password is not a "trivial" word. A trivial word is any value stored in the `uid`, `cn`, `sn`, `givenName`, `ou`, or `mail` attribute of the user's entry.

By default, password syntax checking is turned off.

Password Length

The Directory Server allows you to specify a minimum length for user passwords. In general, shorter passwords are easier to crack. You can require passwords that are from 2 to 512 characters. A good length for passwords is 8 characters. This is long enough to be difficult to crack, but short enough that users can remember the password without writing it down.

By default, no minimum password length is set.

Password Minimum Age

You can configure the Directory Server to not allow users to change their passwords for time you specify. You can use this feature in conjunction with the `passwordHistory` attribute to discourage users from reusing old passwords.

Setting the password minimum age (`passwordMinAge`) attribute to 2 days, for example, prevents a user from repeatedly changing her password during a single session to cycle through the password history and reuse an old password once it is removed from the history list. You can specify any number from 0 to 24,855 days. A value of zero (0) indicates that the user can change the password immediately.

Password History

You can set up the Directory Server to store from 2 to 24 passwords in history, or, you can disable password history, thus allowing users to reuse passwords.

If you set up your password policy to enable password history, the directory stores a specific number of old passwords. If a user attempts to reuse one of the passwords the Directory Server has stored, the directory rejects the password. This feature prevents users from reusing a couple of passwords that are easy to remember.

The passwords remain in history even if you turn the history feature off. This means that if you turn the password history option back on, users cannot reuse the passwords that were in the history before you disabled password history.

The server does not maintain a password history by default.

Password Storage Scheme

The password storage scheme specifies the type of encryption used to store Directory Server passwords within the directory. You can specify:

- Clear text (no encryption)
- Secure Hash Algorithm (SHA)
- Salted Secure Hash Algorithm (SSHA). This encryption method is the default.
- UNIX crypt algorithm

Although passwords stored in the directory can be protected through the use of access control information (ACI) instructions, it is still not a good idea to store cleartext passwords in the directory. The crypt algorithm provides compatibility with UNIX passwords. SSHA is the most secure of the choices.

Designing a Password Policy in a Replicated Environment

Password and account lockout policies are enforced in a replicated environment as follows:

- Password policies are enforced on the data master.
- Account lockout is enforced on the replicas.

The password policy information in your directory, such as password age, the account lockout counter, and the expiration warning counter, are all replicated. However, the configuration information is kept locally and is not replicated. This information includes the password syntax and the history of password modifications.

When configuring a password policy in a replicated environment, consider the following points:

- All replicas issue warnings of an impending password expiration. This information is kept locally on each server, so if a user binds to several replicas in turn, the user receives the same warning several times. In addition, if the user changes the password, it may take time for this information to filter to the replicas. If a user changes a password and then immediately rebinds, the bind may fail until the replica registers the changes.
- You want the same bind behavior to occur on all servers, including masters and replicas. Make sure you create the same password policy configuration information on each server.
- Account lockout counters may not work as expected in a multi-master environment.

Designing an Account Lockout Policy

Once you have established a password policy for your directory, you can protect your user passwords from potential threats by configuring an account lockout policy.

The lockout policy works in conjunction with the password policy to provide further security. The account lockout feature protects against hackers who try to break into the directory by repeatedly trying to guess a user’s password. You can set up your password policy so that a specific user is locked out of the directory after a given number of failed attempts to bind.

Designing Access Control

Once you decide on one or more authentication schemes to establish the identity of directory clients, you need to decide how to use the schemes to protect information contained in your directory. Access control allows you to specify that certain clients have access to particular information, while other clients do not.

You specify access control using one or more access control list (ACL). Your directory's ACLs consist 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 following:

- 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, for all users belonging to a specific group, or for all users of the directory. Lastly, you can define access for a network location such as an IP address or a DNS name.

About the ACI Format

When designing your security policy, it is helpful to understand how ACIs are represented in your directory. It is also helpful to understand what permissions you can set in your directory. This section gives you a brief overview of the ACI mechanism. For a complete description of the ACI format, see the *iPlanet Directory Server Administrator's Guide*.

Directory ACIs take the following general form:

target permission bind_rule

The ACI variables are defined below:

- *target*

Specifies the entry (usually a subtree) the ACI targets, the attribute it targets, or both. 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 application, then the ACI applies to the client application.

So, ACIs are expressed 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*) . . .

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 the part that states “if you bind as Babs Jensen.” The target is Babs Jensen’s phone number, and the permission is write access.

Targets

You must decide what entry is targeted by every ACI you create in your directory. If you target a directory entry that is 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 140.

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 a particular room.

- Compare

  Indicates whether the data may be used in comparison operations. Compare implies the ability to search, but actual directory information is not returned because of the search. Instead, a simple Boolean value is returned that indicates whether the compared values match. This is used to match `userPassword` attribute values during directory authentication.

- Selfwrite

  Used only for group management. This permission allows someone to add to or delete themselves 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.

- Proxy

  Indicates that the user can use any other DN, except Directory Manager, to access the directory with the rights of this DN.

Bind Rules

The bind rule usually indicates the bind DN subject to the permission. It can also specify bind attributes such as time of day or IP address.

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.

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 is 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 requesting access, then the bind rule is true. For example, you can grant specific permission that allows individuals to update their own entries.

- 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.

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 about how to set ACIs in your directory, see the *iPlanet Directory Server Administrator's Guide*.

The Precedence Rule

When a user attempts any kind of access to a directory entry, Directory Server examines the access control set in the directory. To determine access, Directory Server applies the Precedence Rule. The rule states that when two conflicting permissions exist, 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. 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 that user. Then you have to create an additional allow-for-write permission for the user in question.

Allowing or Denying Access

You can explicitly allow or deny access to your directory tree. Be careful of explicitly denying access to the directory. Because of the precedence rule, if the directory finds rules explicitly forbidding access, the directory will forbid access regardless of any conflicting permissions that may grant access.

Limit the scope of your allow access rules to include only the smallest possible subset of users or client applications. 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. Alternatively, you can 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 you avoid the need to set an explicit deny privilege.

When to Deny Access

You rarely need to set an explicit deny. However, you may find an explicit deny useful in the following circumstances:

- 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 only adds to your administrative burden. As soon as possible, 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.

If you are allowing a person or group of people to manage some part of the directory tree, but you want to make sure that they do not modify some aspect of the tree, use an explicit deny. For example, if you want to make sure the Mail Administrators do not allow write access to the common name attribute, then set an 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. Often administrators place access control rules on entries of type `country`, `organization`, `organizationalUnit`, `inetOrgPerson`, or `group`.

To simplify your ACL administration, group your rules 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. LDAP search filters allows you to set 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.

Filtered access control rules let you use predefine levels of access. For example, suppose your directory contains home address and telephone number information. Some people want to publish this information, while others 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 (meaning enabled). 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 about using LDAP search filters, and on using LDAP search filters with ACIs, see the *iPlanet Directory Server Administrator's Guide.*

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 here to provide you with a complete list.

- Minimize the number of ACIs in your directory.

  Although Directory Server can evaluate over 50,000 ACIs, it is difficult to manage a large number of ACI statements. A large number of ACIs makes it hard for you to determine immediately the directory object available to particular clients.

  iPlanet Directory Server 5.0 provides a new feature that minimizes the number of ACIs in the directory by using macros. Macros are placeholders that are used to represent a DN, or a portion of a DN, in an ACI. You can use the macro to represent a DN in the target portion of the ACI, or in the bind rule portion, or both. For more information on macro ACIs, refer to "Managing Access Control" in the *iPlanet Directory Server Administrator's Guide*.

- Balance allow and deny permissions.

  Although the default rule is to deny access to any user who has not been specifically granted access, you might find that you can save on the number of ACIs by using one ACI allowing access close to the root of the tree, and a small number of deny ACIs close to the leaf entries. This scenario can avoid the use of multiple allow ACIs close to the leaf entries.

- 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 contains dozens of 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 in ACIs, run an `ldapsearch` operation using the same filter to make sure you know what the results of the changes mean to your directory.

- 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 becomes increasingly easy to accidentally overlap ACIs in this manner. By avoiding ACI overlap, you 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, giving each ACI a short, meaningful name helps you to manage your security model, especially when examining your ACIs from the Directory console.

- 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. Grouping ACIs helps you manage your total list of ACIs, as well as helps you keep the total number of ACIs in your directory to a minimum.

- Avoid using double negatives, such as deny write if the bind DN is not equal to `cn=Joe`.

  Although this syntax is perfectly acceptable for the server, it’s confusing for a human administrator.

Securing Connections With SSL

After designing your authentication scheme for identified users and your access control scheme for protecting information in your directory, you need to design a way to protect the integrity of the information passed among servers and client applications.

To provide secure communications over the network you can use the LDAP protocol over the Secure Sockets Layer (SSL).

SSL can be used in conjunction with the RC2 and RC4 encryption algorithms from RSA. The encryption method selected for a particular connection is the result of a negotiation between the client application and Directory Server.

SSL can also be used in conjuction with CRAM-MD5, which is a hashing mechanism that guarantees that information has not been modified during transmission.

Directory Server can have SSL-secured connections and non SSL connections simultaneously.

For information about enabling SSL, refer to the *iPlanet Directory Server Administrator's Guide*.

Other Security Resources

For more information about designing a secure directory, take a look at the following:

- Netscape Security Notes
  http://home.netscape.com/security/notes/
- *Understanding and Deploying LDAP Directory Services.*
  T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999.
- SecurityFocus.com
  http://www.securityfocus.com/
- Computer Emergency Response Team (CERT) Coordination Center
  http://www.cert.org
- CERT Security Improvement Modules
  http://www.cert.org/security-improvement/

Chapter 8

Directory Design Examples

How you design your directory depends upon the size and nature of your enterprise. This chapter provides examples of how a directory can be applied within a variety of different settings. You can use these examples as a starting point for developing your own directory deployment plan.

This chapter contains the following example directory designs:

- An Enterprise
- A Multinational Enterprise and its Extranet

An Enterprise

Siroe Corporation, an automobile parts manufacturer, is a small company that consists of 500 employees. Siroe decides to deploy Directory Server to support the directory-enabled applications it uses. Siroe's directory design process involves the following steps:

- "Data Design," on page 148
- "Schema Design," on page 148
- "Directory Tree Design," on page 149
- "Topology Design," on page 150
- "Replication Design," on page 153
- "Security Design," on page 155
- "Tuning and Optimizations," on page 156
- "Operations Decisions," on page 157

Data Design

Siroe first decides the type of data it will store in the directory. To do this, Siroe creates a deployment team that performs a site survey to determine how the directory will be used. The deployment team determines the following:

- Siroe's directory will be used by iPlanet Messaging Server, iPlanet Web Server, iPlanet Calendar Server, a human resources application, and a white pages application.
- iPlanet Messaging Server does exact searches on attributes such as the `uid`, `mailServerName`, and `mailAddress` attributes. To improve database performance, Siroe will maintain indexes for these attributes to support searches by iPlanet Messaging Server.

  For more information on using indexes refer to "Using Indexes to Improve Database Performance," on page 94.
- The white pages application will frequently search for user names and phone numbers. The directory will therefore need to be able to handle lots of substring, wildcard, and sounds-like searches which return large sets of results. Siroe decides to maintain approximate and substring indexes for the `dn`, `sn`, and `givenName` attributes
- Siroe's directory maintains user and group information to support an iPlanet server-based intranet deployed throughout the organization. Most of Siroe's user and group information will be centrally managed by a group of directory administrators. However, Siroe also wants email information to be managed by a separate group of mail administrators.
- Siroe plans to support public key infrastructure (PKI) applications in the future, such as s/mime email, so needs to be prepared to store the public key certificates of users in the directory.

Schema Design

Siroe's deployment team decides to use the `inetOrgPerson` object class to represent the entries in the directory. This object class is appealing because it allows the `userCertificate` and `uid(userID)` attributes, both of which are needed by the applications supported by Siroe's directory.

Siroe also wants to customize the default directory schema. Siroe creates the `siroePerson` object class to represent employees of Siroe Corporation. It derives this object class from the `inetOrgPerson` object class.

The siroePerson object class allows one attribute, the siroeID attribute. This attribute contains the special employee number assigned to each Siroe employee.

In the future, Siroe can add new attributes to the siroePerson object class as needed.

Directory Tree Design

Siroe creates a directory tree as follows:

- The root of the directory tree is Siroe's internet domain name: dc=siroe,dc=com.
- The directory tree has four branch points: ou=people, ou=groups, ou=roles and ou=resources.
- All of Siroe's people entries are created under the ou=people branch.

  The people entries are all members of the person, organizationalPerson, inetOrgPerson, and siroePerson object classes. The uid attribute uniquely identifies each entry's DN. For example, Siroe contains entries for Babs Jensen (uid=bjensen) and Emily Stanton (uid=estanton).
- Siroe creates three roles, one for each department in Siroe: sales, marketing, and accounting.

  Each person entry contains a role attribute which identifies the department to which the person belongs. Siroe can now create ACIs based upon these roles.

  For more information about roles, refer to "About Roles," on page 71.
- Two group branches are created under the ou=groups branch.

  The first group, cn=administrators, contains entries for the directory administrators that manage the directory contents.

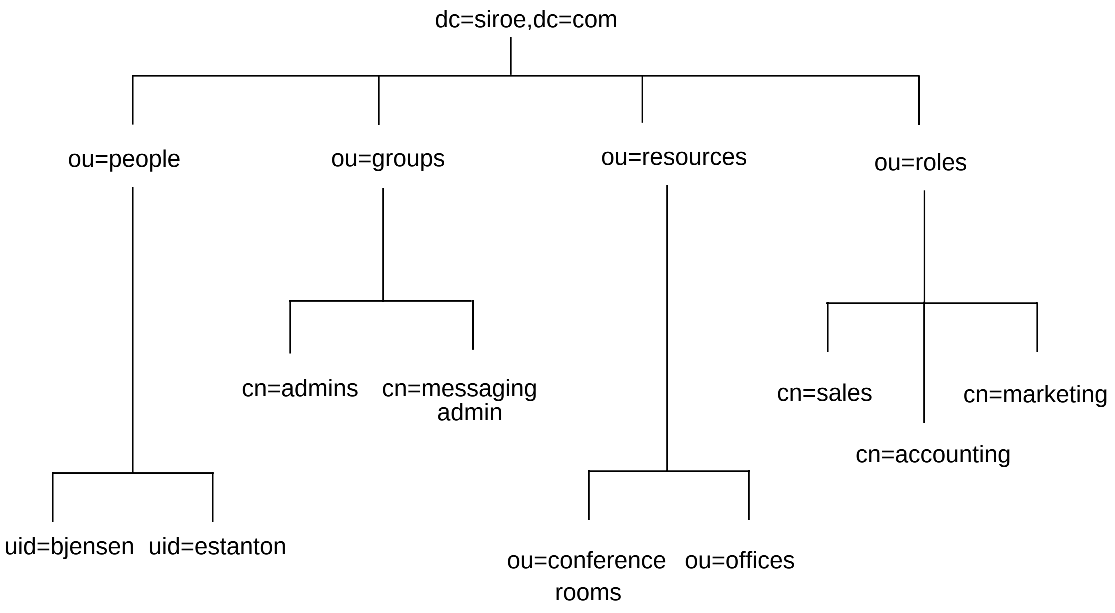
  Mail administrators use the second group, cn=messaging admin, to manage mail accounts. This group corresponds to the administrators group used by iPlanet Messaging Server. Siroe makes sure that the group it configures for Messaging Server is different from the group it creates for Directory Server.
- Two branches are created under the ou=resources branch, one for conference rooms (ou=conference rooms) and one for offices (ou=offices).
- Siroe creates a class of service (CoS) that provides values for the mailquota attribute depending upon whether or not an entry belongs to the administrative group.

This CoS gives administrators a mail quota of 500 MB while ordinary Siroe employees have a mail quota of 100 MB.

For more information about CoS, refer to “About Class of Service,” on page 72.

The following diagram illustrates the directory tree resulting from the design steps listed above:

Figure 8-1 Directory Tree for Siroe Corporation



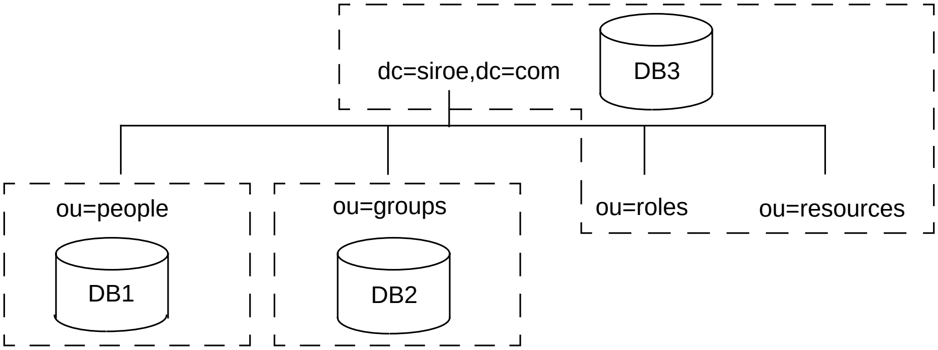
Topology Design

Next, Siroe designs both its database and server topologies. The following sections describe each topology in detail.

Database Topology

Siroe designs a database topology in which the people branch is stored in one database (DB1), the groups branch is stored in another database (DB2), and the resources branch, roles branch, and the root suffix information are stored in a third database (DB3). The database topology for Siroe’s directory looks as follows:

Figure 8-2 Database Topology for Siroe Corporation



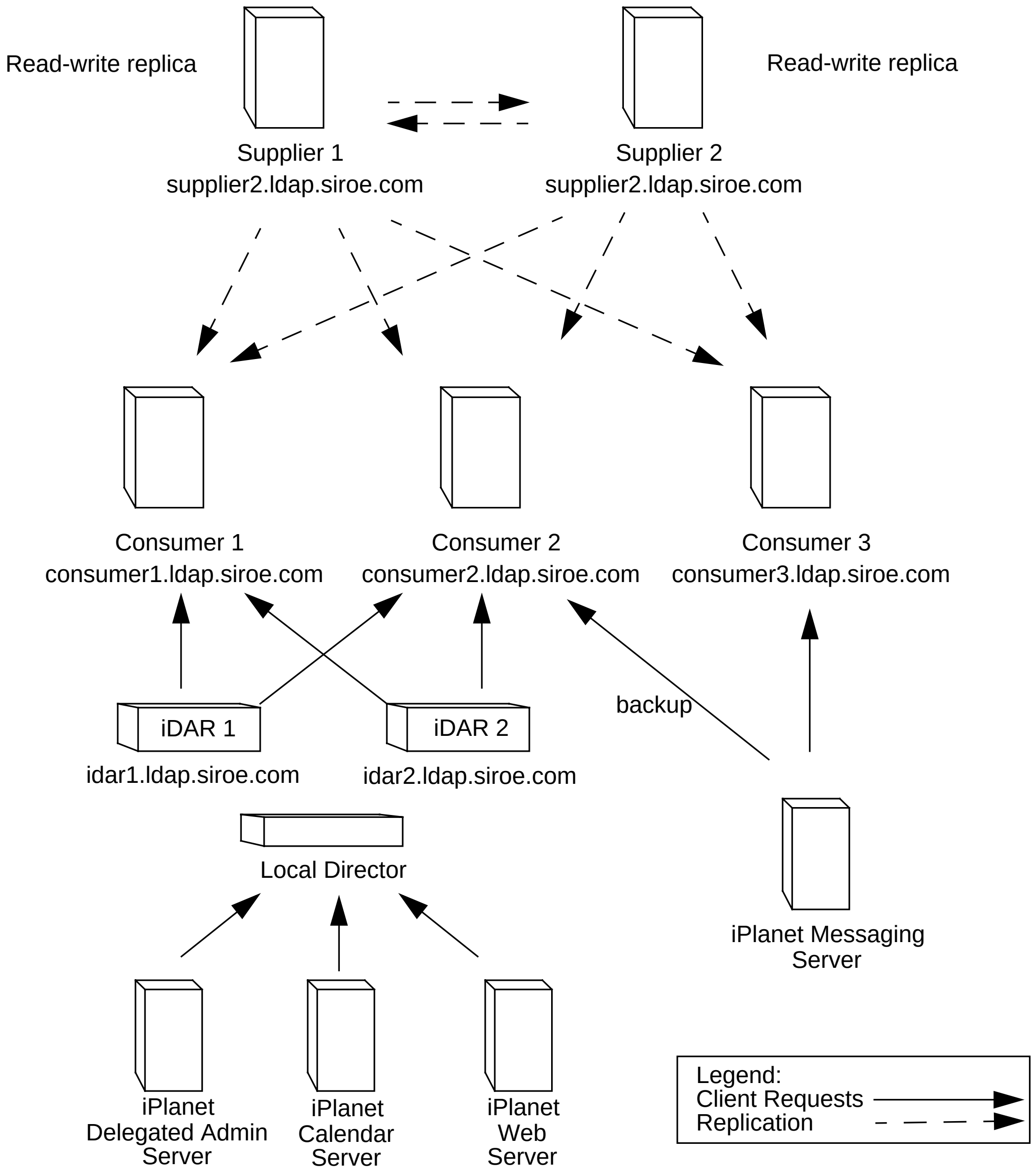
Server Topology

Each of the two supplier servers updates all three consumer servers in Siroe's deployment of directory server. These consumers supply data to one iPlanet Messaging Server and to the other Unified User Management products using iPlanet Data Access Router (iDAR).

For more information about the iPlanet Unified User Management products, refer to http://www.iplanet.com/products/.

The Siroe server topology follows:

Figure 8-3 Server Topology for Siroe Corporation



Modify requests from the iPlanet servers (such as the Calendar Server or the Delegated Admin Server) are routed by iDAR to the appropriate consumer server. The consumer server uses smart referrals to route the request to the supplier server responsible for the master copy of the piece of data being modified.

Replication Design

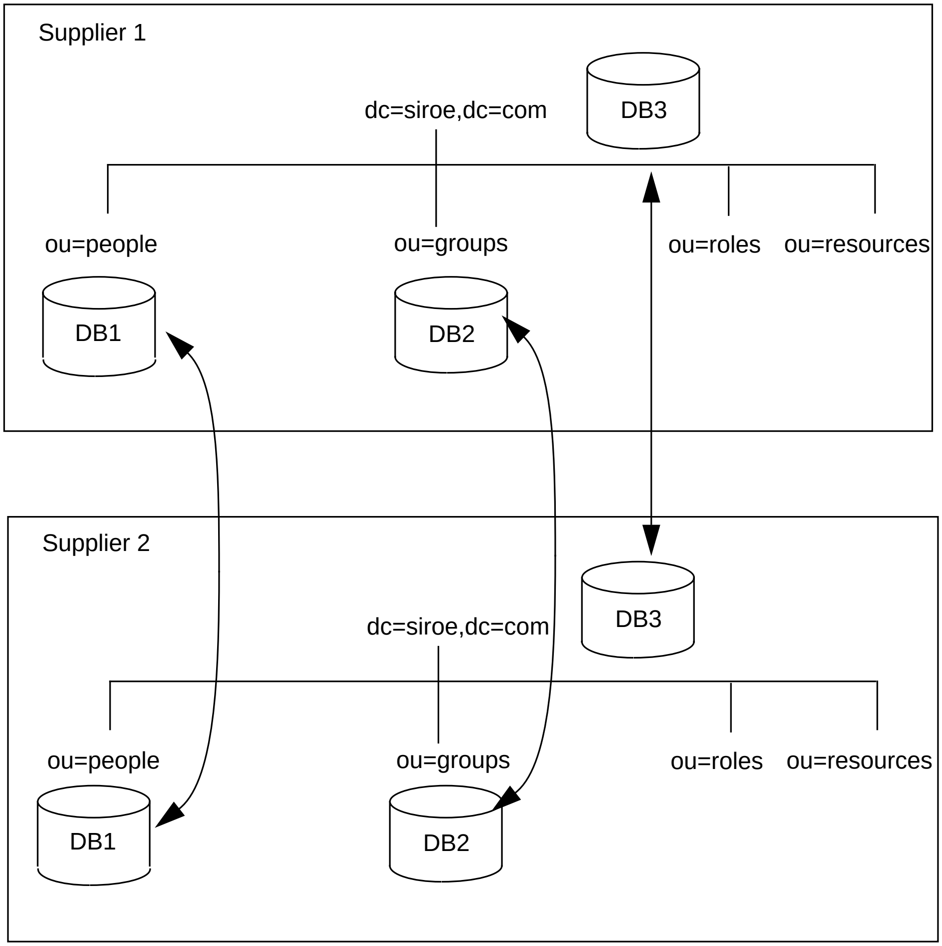
Siroe decides to use a multi-master replication design to ensure the high availability of its directory data. For more information about multi-master replication, refer to “Multi-Master Replication,” on page 103.

The following sections provide more details about the supplier server architecture and the supplier-consumer server topology.

Supplier Architecture

Siroe uses two supplier servers in a multi-master replication architecture. The suppliers update one another so that the directory data remains consistent. The supplier architecture works as follows:

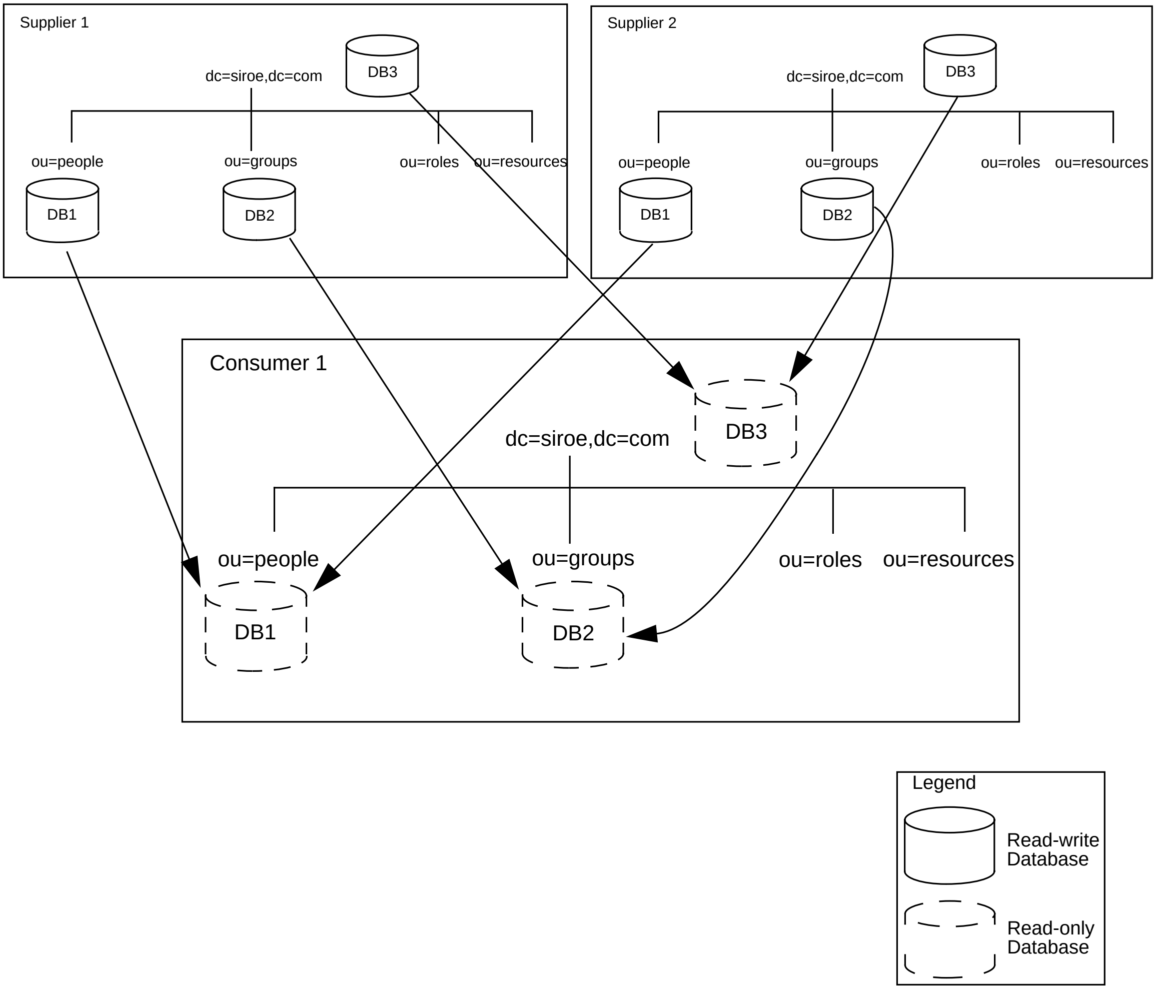
Figure 8-4 Supplier Architecture for Siroe Corporation



Supplier Consumer Architecture

The following diagram describes how the supplier servers replicate to each consumer in the Siroe deployment of the directory:

Figure 8-5 Supplier/Consumer Architecture for Siroe Corporation



Each of the three consumer servers is updated by the two supplier servers as shown in Figure 8-5. This ensures that the consumers will not be affected if there is a failure in one of the supplier servers.

Security Design

Siroe decides on the following security design to protect its directory data:

- Siroe creates an ACI that allows employees to modify their own entries.

  Users can modify all attributes except the `uid`, `manager` and `department` attributes.

- To protect the privacy of employee data, Siroe develops an ACI that allows only the employee and an employee's manager to see the employee's home address and phone number.

- Siroe creates an ACI that allows the two administrator groups the appropriate directory permissions is created at the root of the directory tree.

  The directory administrators group needs full access to the directory. The 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. Siroe also grants the messaging administrators group `write`, `delete`, and `add` permissions to the group subdirectory for creation of mail groups.

- A general access control is created on the root of the directory tree that allows anonymous access for read, search, and compare access.

  This ACI denies anonymous users access to password information.

- To protect the server from denial of service attacks and inappropriate use, Siroe sets resource limits based on the DN used by directory clients to bind.

  Siroe allows anonymous users to receive 100 entries at a time in response to search requests, administrator users to receive 1,000 entries, and system administrators to receive an unlimited number of entries. For more information about setting resource limits based on the bind DN, refer to "User Account Management" in the *iPlanet Directory Server Administrator's Guide*.

- Siroe creates a password policy where passwords must be at least 8 characters in length and expire after 90 days.

  For more information about password policies, refer to "Designing a Password Policy," on page 131.

- Siroe creates an ACI that gives members of the accounting role access to all payroll information.

Tuning and Optimizations

Siroe optimizes its deployment of directory by doing the following:

- Running the `idsktune` utility

The `idsktune` utility provides an easy and reliable way of checking the patch levels and kernel parameter settings for your system. For more information about `idsktune`, see the *iPlanet Directory Server Installation Guide*.

- Optimizing the entry and database caches

  Siroe sets the entry cache to 2000 entries, and the database cache to 250 MB to ensure that all of the indexes fit into RAM, optimizing server performance.

Operations Decisions

Siroe makes the following decisions regarding the day-to-day operation of its directory:

- Back up the databases every night and write the backups to tape once a week.
- Use SNMP to monitor the server status.

  For more information about SNMP, refer to the *iPlanet Directory Server Administrator's Guide*.
- Auto-rotate the access logs.
- Monitor the error log to see if the server is performing as expected.
- Monitor the access log to screen for searches that should be indexed.

For more information about the access, error, and audit logs, refer to "Monitoring Server and Database Activity" in the *iPlanet Directory Server Administrator's Guide*.

A Multinational Enterprise and its Extranet

This example builds a directory infrastructure for Siroe International. Siroe Corporation from the previous example has grown into a large, multinational company. This example builds on the directory structure created in the last example for Siroe Corporation, expanding the directory design to meet its new needs.

Siroe has grown into an organization dispersed over three main geographic locations: the US, Europe, and Asia. Siroe now has more than 20,000 employees, all of which live and work in the countries where the Siroe offices are located. Siroe decides to launch a company-wide LDAP directory to improve internal communication, to make it easier to develop and deploy web applications, and to increase security and privacy.

Designing a directory tree for an international corporation involves determining how to logically collect directory entries, how to support data management, and how to support replication on a global scale.

In addition, Siroe wants to create an extranet for use by its parts suppliers and trading partners. An *extranet* is an extension of an enterprise’s intranet to external clients.

The following sections describe the steps in the process of deploying a multinational directory service and extranet for Siroe International:

- “Data Design,” on page 158
- “Schema Design,” on page 159
- “Directory Tree Design,” on page 159
- “Topology Design,” on page 161
- “Replication Design,” on page 165
- “Security Design,” on page 168

Data Design

Siroe International creates a deployment team to perform a site survey. The deployment team determines the following from the site survey:

- iPlanet Messaging Server is used to provide electronic mail routing, delivery, and reading services for most of Siroe’s sites. Enterprise server is used to provide document publishing services. All servers run on the Solaris UNIX operating system.
- Siroe needs to allow data to be managed locally. For example, the European site will be responsible for managing the Europe branch of the directory. This also means that Europe will be responsible for the master copy of its data.
- Because of the geographic distribution of Siroe’s offices, the directory needs to be available to users and applications 24 hours a day.
- Many of the data elements need to accommodate data values of several different languages and characters sets.

The deployment team also determines the following about the data design of the extranet:

- Suppliers will need to log in to Siroe's directory to manage their contracts with Siroe. Suppliers will depend upon data elements used for authentication, such as name and user password.
- Siroe's partners will use the directory as a way to look up the email addresses and phone number of people in the partner network.

Schema Design

Siroe builds upon its original schema design by adding schema elements to support the extranet. Siroe adds two new objects, the `siroeSupplier` object class and the `siroePartner` object class.

The `siroeSupplier` object class allows one attribute, the `siroeSupplierID` attribute. This attribute contains the unique ID assigned by Siroe International to each auto part supplier it works with.

The `siroePartner` object class allows one attribute, the `siroePartnerID` attribute. This attribute contains the unique ID assigned by Siroe International to each trade partner.

For information about customizing the default directory schema, refer to "Customizing the Schema," on page 46.

Directory Tree Design

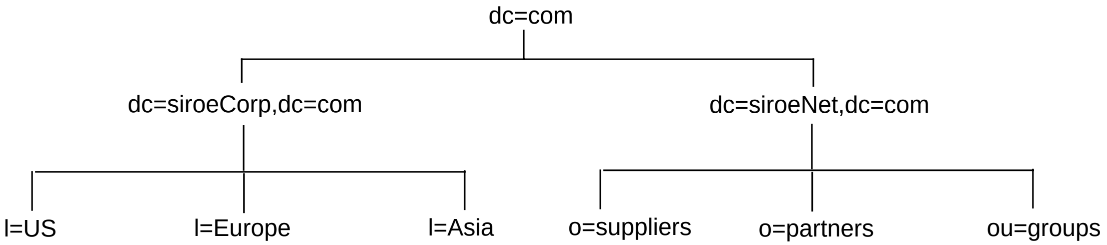
Siroe creates a directory tree as follows:

- The directory tree is rooted in the suffix `dc=com`. Under this suffix, Siroe creates two branch. One branch, `dc=siroeCorp,dc=com`, contains data internal to Siroe International. The other branch, `dc=siroeNet,dc=com`, contains data for the extranet.
- The directory tree for the intranet (under `dc=siroeCorp,dc=com`) has three main branches, each corresponding to one of the regions where Siroe has offices. These branches are identified using the `l(locality)` attribute.
- Each main branch under `dc=siroeCorp,dc=com` mimics the original directory tree design of Siroe Corporation. Under each locality, Siroe creates an `ou=people`, an `ou=groups`, an `ou=roles`, and an `ou=resources` branch. See "Directory Tree for Siroe Corporation," on page 150 for more information about this directory tree design.

- Under the dc=siroeNet,dc=com branch, Siroe creates three branches. One branch for suppliers (o=suppliers), one branch for partners (o=partners), and one branch for groups (ou=groups).
- The ou=groups branch of the extranet contains entries for the administrators of the extranet as well as mailing lists that partners subscribe to for up-to-date information on auto part manufacturing.

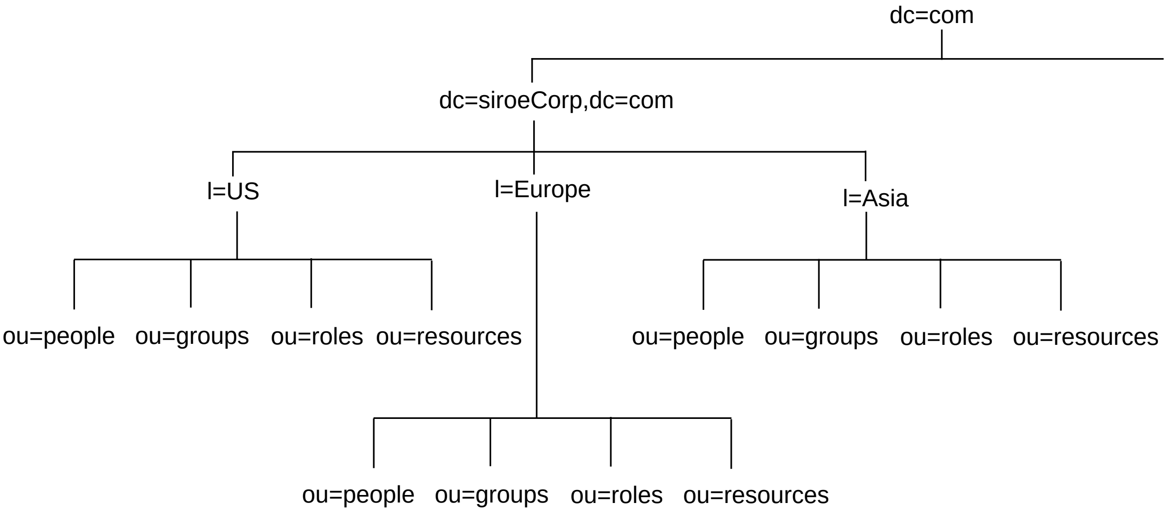
The basic directory tree that results appears as follows:

Figure 8-6 Basic Directory Tree for Siroe International



The directory tree for the Siroe intranet appears as follows:

Figure 8-7 Directory Tree for Siroe International's Intranet

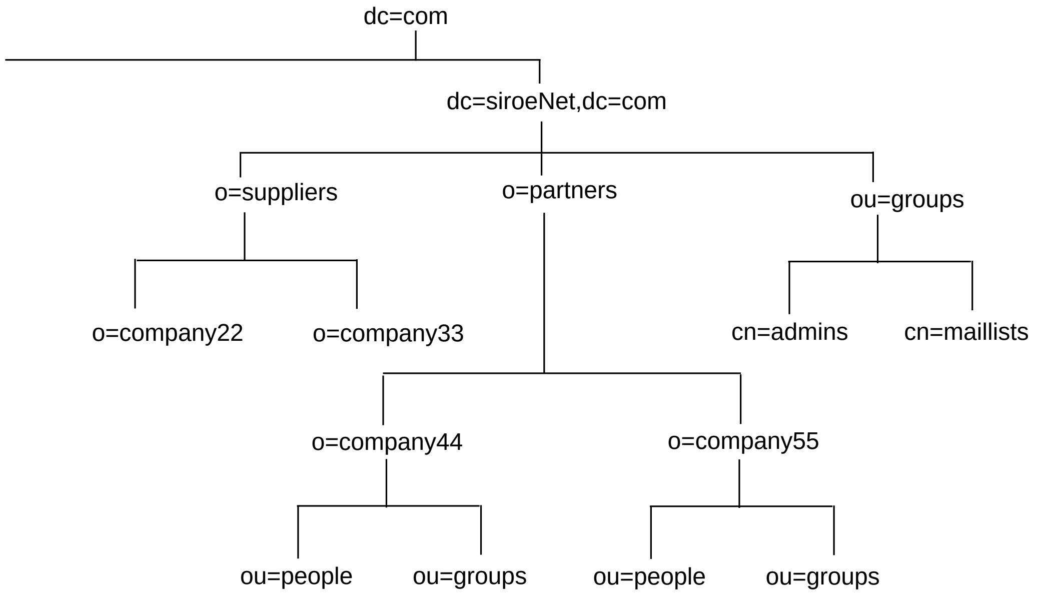


The entry for the l=Asia entry appears in LDIF as follows:

```
dn: l=Asia,dc=siroeCorp,dc=com
objectclass: top
objectclass: locality
l: Asia
description: includes all sites in Asia
```

The directory tree for Siroe's extranet appears as follows:

Figure 8-8 Directory Tree for Siroe International's Extranet



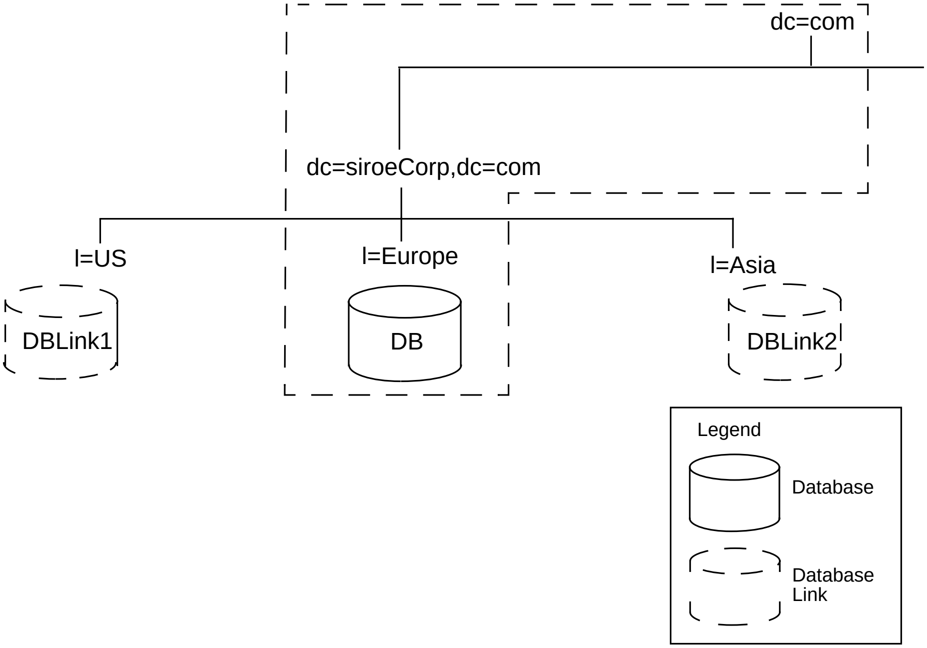
Topology Design

Next, Siroe designs both its database and server topologies. The following sections describe each topology in more detail.

Database Topology

The following diagram illustrates the database topology of one of Siroe's main localities, Europe:

Figure 8-9 Database Topology for Siroe Europe

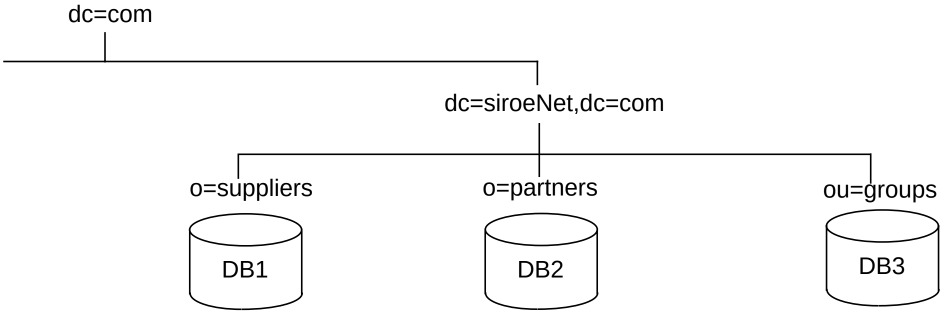


The database links point to databases stored locally in each country. For example, operation requests received by the Siroe Europe server for the data under the `l=US` branch are chained by a database link to a database on a server in Austin, Texas. For more information about database links and chaining, refer to "Using Chaining," on page 89.

The master copy of the data for `dc=siroeCorp,dc=com` and the root entry, `dc=com`, is kept in the Europe, as shown by the box in Figure 8-9.

The data center in Europe contains the master copies of the data for the extranet. The extranet data is stored in three databases, one for each of the main branches. The following figures shows the database topology for the extranet:

Figure 8-10 Database Topology for Siroe International's Extranet



As illustrated in Figure 8-10, the master copy of the data for `o=suppliers` is stored in database one, the master copy of the data for `o=partners` is stored in database two, and the master copy of the data for `ou=groups` is stored in database three.

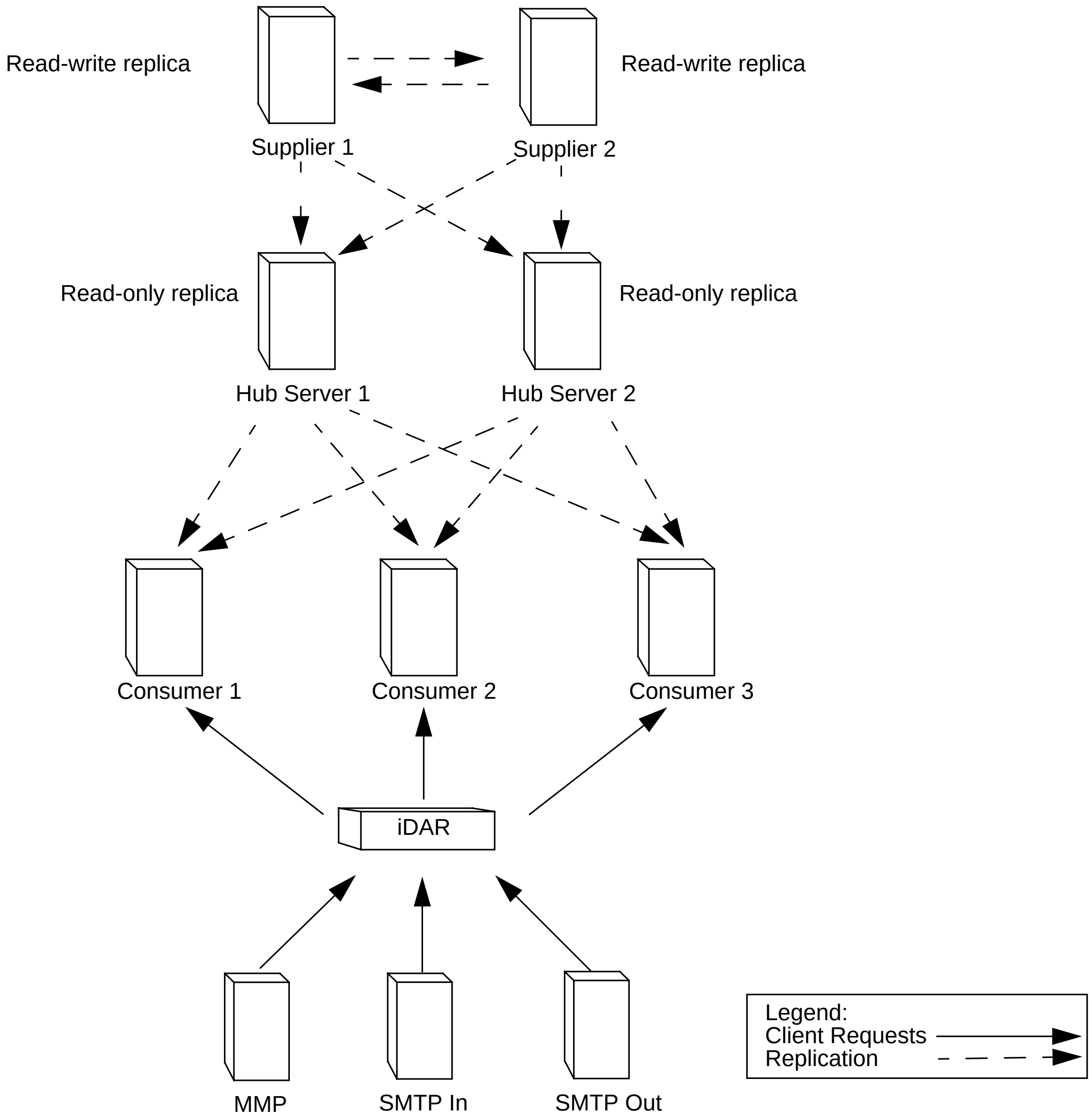
Server Topology

Siroe develops two server topologies, one for the corporate intranet and one for the partner extranet.

For the intranet, Siroe decides to have a master database for each major locality. This means it has three data centers, each containing two supplier servers, two hub servers, and three consumer servers.

The architecture of Siroe Europe's data center appears as follows:

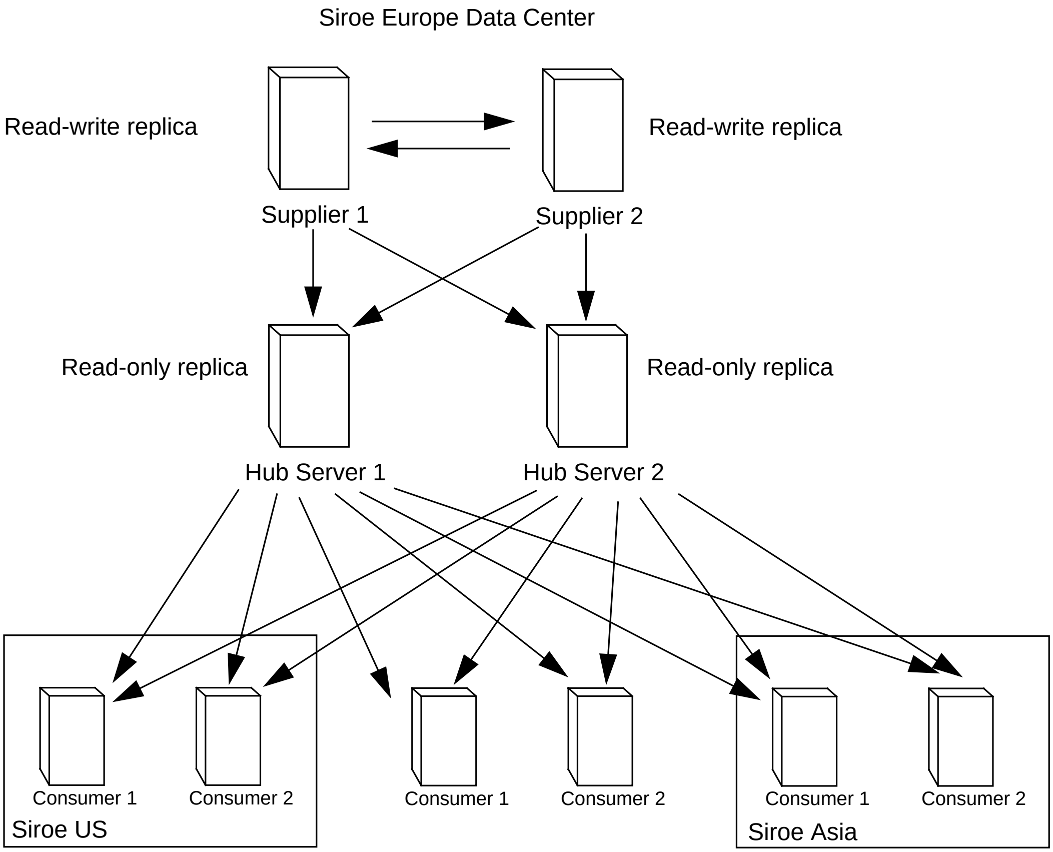
Figure 8-11 Server Topology for Siroe Europe



Siroe's extranet data is mastered in Europe. This data is replicated to two consumer servers in the US data center and two consumer servers in the Asia data center. In all, Siroe requires ten servers to support the extranet.

The server architecture of Siroe's extranet appears as follows in the Siroe Europe data center:

Figure 8-12 Server Topology for Siroe International's Extranet



The hub servers replicate the data to two consumer servers in the Siroe Europe data center, two consumer servers in the Siroe US data center, and two consumer servers in the Siroe Asia data center.

Replication Design

Siroe considers the following when designing replication for its directory:

- Data will be managed locally.

- Quality of network connections varies from site to site.
- Database links will be used to connect data on remote servers.
- Hub servers that contain read-only copies of the data will be used to replicate data to consumer servers.

  The hub servers are located near important directory-enable applications such as a mail server or a web server.

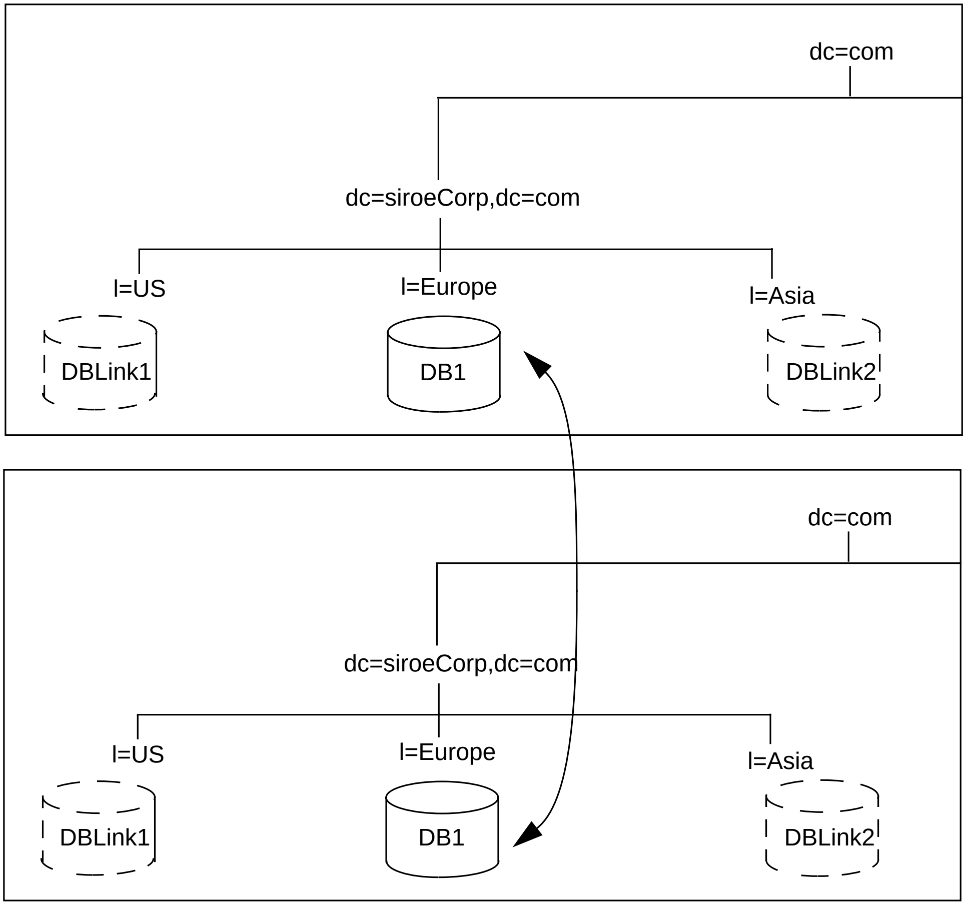
  Hub servers remove the burden of replication from the supplier servers, so the suppliers can concentrate on doing write operations. In the future, as Siroe expands and needs to add more consumer servers, the additional consumers do not affect the performance of the suppliers.

  For more information about hub servers, refer to “Cascading Replication,” on page 105.

Supplier Architecture

For the Siroe intranet, each locality keeps the master copy of its data and uses database links to chain to the data of other localities. For the master copy of its data, each locality uses a multi-master replication architecture. For example, the supplier architecture for Europe, which includes the `dc=siroeCorp,dc=com` and `dc=com` information, appears as follows:

Figure 8-13 Supplier Architecture for Siroe Europe

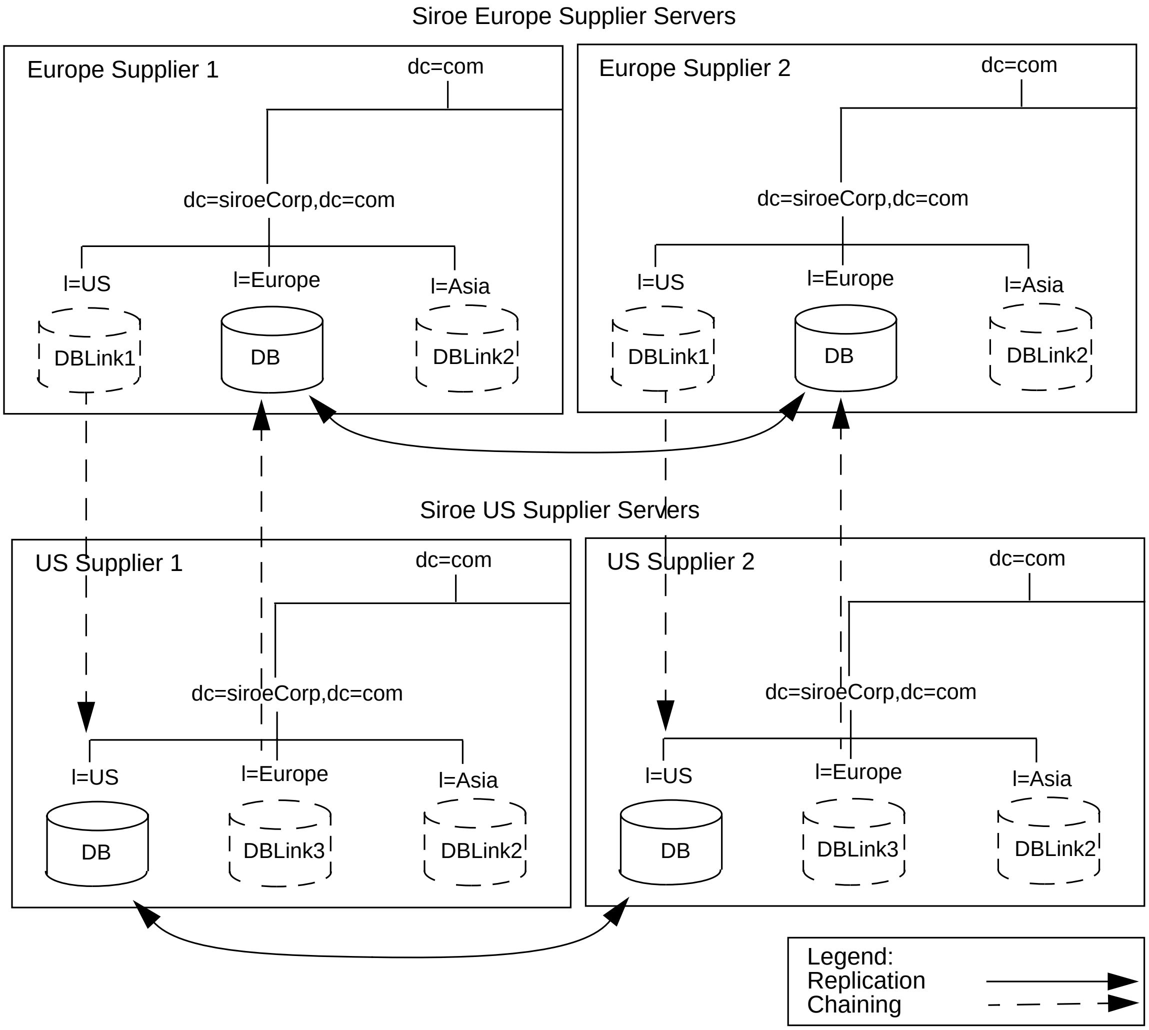


Each locality contains two suppliers, which share master copies of the data for that site. Each locality is therefore responsible for the master copy of its own data. Using a multi-master architecture ensures the availability of the data and helps balance the load of work managed by each supplier server.

To reduce the risk of total failure, Siroe Corporation uses multiple read-write master directory servers at each site.

The following diagram illustrates the interaction between the two supplier servers in Europe and the two supplier servers in the US:

Figure 8-14 Supplier/Supplier Architecture for Siroe Europe and Siroe US



The same relationship as illustrated in Figure 8-14 exists between Siroe US and Siroe Asia, and between Siroe Europe and Siroe Asia.

Security Design

Siroe International builds upon its previous security design, adding the following access controls to support its new multinational intranet:

- Siroe adds general ACIs to the root of the intranet, creating more restrictive ACIs in each country and the branches beneath each country.
- Siroe decides to use macro ACIs to minimize the number of ACIs in the directory.

  Siroe uses a macro to represent a DN in the target or bind rule portion of the ACI. When the directory gets an incoming LDAP operation, the ACI macros are matched against the resource targeted by the LDAP operation. If there is a match, the macro is replaced by the value of the DN of the targeted resource.

  For more information about macro ACIs, refer to the *iPlanet Directory Server Administrator's Guide*.

Siroe adds the following access controls to support its extranet:

- Siroe decides to use certificate-based authentication for all extranet activities. When people log in to the extranet, they need a digital certificate. The directory is used to store the certificates. Because the directory stores the certificates, users can send encrypted email by looking up public keys stored in the directory.
- Siroe creates an ACI that forbids anonymous access to the extranet. This protects the extranet from denial of service attacks.
- Siroe wants updates to the directory data to come only from a Siroe hosted application. This means that partners and suppliers using the extranet can only use the tools provided by Siroe. Restricting extranet users to Siroe's preferred tools allows Siroe administrators to use the audit logs to track the use of the directory and limits the types of problems that can be introduced by extranet users outside of Siroe International.
- Siroe will use iDAR to add additional security. For more information about iDAR, go to http://www.iplanet.com/.

Glossary

access control instruction *See ACI.*

ACI Access Control Instruction. An instruction that grants or denies permissions to entries in the directory.

access control list *See ACL.*

ACL Access control list. The mechanism for controlling access to your directory.

access rights In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.

account inactivation Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.

All IDs Threshold A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token.

All IDs token A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.

anonymous access When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index Allows for efficient approximate or “sounds-like” searches.

attribute Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute value.

attribute list A list of required and optional attributes for a given entry type or object class.

authenticating directory server In pass-through authentication (PTA), the authenticating directory server is the directory server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the host.

authentication (1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.

(2) Allows a client to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

authentication certificate Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.

base DN Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.

base distinguished name *See base DN.*

bind DN Distinguished name used to authenticate to Directory Server when performing an operation.

bind distinguished name *See bind DN.*

bind rule In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.

branch entry An entry that represents the top of a subtree in the directory.

browser Software, such as Netscape Navigator, used to request and view World Wide Web material stored as HTML files. The browser uses the HTTP protocol to communicate with the host server.

browsing index Otherwise known as the virtual view index, speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branchpoint in the directory tree to improve display performance.

CA *See Certificate Authority.*

cascading replication In a cascading replication scenario, one server, often called the hub supplier acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a change log. It receives updates from the supplier server that holds the master copy of the data, and in turn supplies those updates to the consumer.

certificate A collection of data that associates the public keys of a network user with their DN in the directory. The certificate is stored in within the directory as user object attributes.

Certificate Authority Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. Also known as a CA.

CGI Common Gateway Interface. An interface for external programs to communicate with the HTTP server. Programs written to use CGI are called CGI programs or CGI scripts, and can be written in many of the common programming languages. CGI programs handle forms or perform output parsing that is not done by the server itself.

chaining A method for relaying requests to another server. Results for the request are collected, compiled and then returned to the client.

change log A change log is record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on consumer servers, or on other masters, in the case of multi-master replication.

character type Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext Encrypted information that cannot be read by anyone without the proper key to decrypt the information.

CIR *See consumer-initiated replication.*

class definition Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.

class of service *See CoS.*

classic CoS A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.

client *See LDAP client.*

code page An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.

collation order Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.

consumer Server containing replicated directory trees or subtrees from a supplier server.

consumer-initiated replication Replication configuration where consumer servers pull directory data from supplier servers.

consumer server In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.

CoS A method for sharing attributes between entries in a way that is invisible to applications.

CoS definition entry Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

CoS template entry Contains a list of the shared attribute values.

daemon A background process on a Unix machine that is responsible for a particular system task. Daemon processes do not need human intervention to continue functioning.

DAP Directory Access Protocol. The ISO X.500 standard protocol that provides client access to the directory.

data master The server that is the master source of a particular piece of data.

database link An implementation of chaining. The database link behaves like a database but has no persistent storage. Instead, it points to data stored remotely.

default index One of a set of default indexes created per database instance. Default indexes can be modified, although care should be taken before removing them, as certain plug-ins may depend on them.

definition entry *See CoS definition entry*.

Directory Access Protocol *See DAP.*

directory tree The logical representation of the information stored in the directory. It mirrors the tree model used by most file systems, with the tree's root point appearing at the top of the hierarchy. Also known as DIT.

Directory Manager The privileged database administrator, comparable to the root user in UNIX. Access control does not apply to the directory manager.

Directory Server Gateway (DSGW) A collection of CGI forms that allows a browser to perform LDAP client functions, such as querying and accessing a Directory Server, from a web browser.

directory service A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

distinguished name String representation of an entry's name and location in an LDAP directory.

DIT *See directory tree.*

DN s*ee distinguished name*.

DM *See Directory Manager.*

DNS Domain Name System. The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as `www.iPlanet.com`). Machines normally get the IP address for a hostname from a DNS server, or they look it up in tables maintained on their systems.

DNS alias A DNS alias is a hostname that the DNS server knows points to a different host—specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as `www.[yourdomain].[domain]` might point to a real machine called `realthing.[yourdomain].[domain]` where the server currently exists.

DSGW *See Directory Server Gateway (DSGW).*

entry A group of lines in the LDIF file that contains information about an object.

entry distribution Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index Allows you to search efficiently for entries containing a specific attribute value.

file extension The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename `index.html` the file extension is `html`.

file type The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter A constraint applied to a directory query that restricts the information returned.

filtered role Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

gateway *See Directory Server Gateway (DSGW).*

general access When granted, indicates that all authenticated users can access directory information.

hostname A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, `www.iPlanet.com` is the machine www in the subdomain iPlanet and com domain.

HTML Hypertext Markup Language. The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Netscape Navigator how to display text, position graphics and form items, and display links to other pages.

HTTP Hypertext Transfer Protocol. The method for exchanging information between HTTP servers and clients.

HTTPD An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an httpd.

HTTP-NG The next generation of Hypertext Transfer Protocol.

HTTPS A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.

hub supplier In the context of replication, a server that holds a replica that is copied from a different server, and in turn replicates it to a third server. See also cascading replication.

index key Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS An indirect CoS identifies the template entry using the value of one of the target entry's attributes.

international index Speeds up searches for information in international directories.

International Standards Organization *See ISO.*

IP address Internet Protocol address. A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10).

ISO International Standards Organization

knowledge reference Pointers to directory information stored in different databases.

LDAP Lightweight Directory Access Protocol. Directory service protocol designed to run over TCP/IP and across multiple platforms.

LDAPv3 Version 3 of the LDAP protocol, upon which Directory Server bases its schema format

LDAP client Software used to request and view LDAP entries from an LDAP Directory Server. See also *browser*.

LDAP Data Interchange Format *See LDAP Data Interchange Format.*

LDAP URL Provides the means of locating directory servers using DNS and then completing the query via LDAP. A sample LDAP URL is `ldap://ldap.iplanet.com`

LDBM database A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.

LDIF LDAP Data Interchange Format. Format used to represent Directory Server entries in text form.

leaf entry An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Lightweight Directory Access Protocol *See LDAP.*

locale Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.

managed object A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.

managed role Allow you to create an explicit enumerated list of members.

management information base *See MIB.*

mapping tree A data structure that associates the names of suffixes (subtrees) with databases.

master agent *See SNMP master agent.*

matching rule Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5 A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data, that is unique with high probability, and is mathematically extremely hard to produce a piece of data that will produce the same message digest.

MD5 signature A message digest produced by the MD5 algorithm.

MIB Management Information Base. All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace Management Information namespace. The means for directory data to be named and referenced. Also called the directory tree.

monetary format Specifies the monetary symbol used by specific region, whether the symbol goes before or after its value, and how monetary units are represented.

multi-master replication An advanced replication scenario in which two servers each hold a copy of the same read-write replica. Each server maintains a change log for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.

multiplexor The server containing the database link that communicates with the remote server.

n + 1 directory problem The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.

name collisions Multiple entries with the same distinguished name.

nested role Allow you to create roles that contain other roles.

network management application Network Management Station component that graphically displays information about SNMP managed devices (which device is up or down, which and how many error messages were received, etc.).

network management station *See NMS.*

NIS Network Information Service. A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, file systems, and network parameters throughout a network of computers.

NMS Network Management Station. Powerful workstation with one or more network management applications installed.

ns-slapd iPlanet's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server. See also slapd.

object class Defines an entry type in the directory by defining which attributes are contained in the entry.

object identifier A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations.

OID *See object identifier.*

operational attribute An operational attribute contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

parent access When granted, indicates that users have access to entries below their own in the directory tree, that is, if the bind DN is the parent of the targeted entry.

pass-through authentication *See PTA.*

pass-through subtree In pass-through authentication, the PTA directory server will pass through bind requests to the authenticating directory server from all clients whose DN is contained in this subtree.

password file A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as `/etc/passwd`, because of where it is kept.

password policy A set of rules that govern how passwords are used in a given directory.

permission In the context of access control, the permission states whether access to the directory information is granted or denied, and the level of access that is granted or denied. See access rights.

PDU Protocol Data Unit. Encoded messages which form the basis of data exchanges between SNMP devices.

pointer CoS A pointer CoS identifies the template entry using the template DN only.

presence index Allows you to search for entries that contain a specific indexed attribute.

protocol A set of rules that describes how devices on a network exchange information.

protocol data unit *See PDU.*

proxy authentication A special form of authentication where the user requesting access to the directory does not bind with its own DN but with a proxy DN.

proxy DN Used with proxied authorization. The proxy DN is the DN of an entry that has access permissions to the target on which the client-application is attempting to perform an operation.

PTA Pass-through authentication. Mechanism by which one directory server consults another to check bind credentials.

PTA directory server In pass-through authentication (PTA), the PTA directory server is the server that sends (passes through) bind requests it receives to the authenticating directory server.

PTA LDAP URL In pass-through authentication, the URL that defines the authenticating directory server, pass-through subtree(s) and optional parameters.

RAM Random access memory. The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.

rc.local A file on Unix machines that describes programs that are run when the machine starts. It is also called `/etc/rc.local` because of its location.

RDN Relative distinguished name. The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name.

referential integrity Mechanism that ensures that relationships between related entries are maintained within the directory.

referral (1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP sever that can process the request.

(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding process is called a referral.

replica A database that participates in replication

read-only replica A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.

read-write replica A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.

relative distinguished name *See RDN.*

replication Act of copying directory trees or subtrees from supplier servers to consumer servers.

replication agreement Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the consumer servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.

RFC Request For Comments. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.

role An entry grouping mechanism. Each role has *members*, which are the entries that possess the role.

role-based attributes Attributes that appear on an entry because it possesses a particular role within an associated CoS template.

root The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.

root suffix The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

schema Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

schema checking Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default and users will receive an error if they try to save an entry that does not conform to the schema.

Secure Sockets Layer *See SSL.*

self access When granted, indicates that users have access to their own entries, that is, if the bind DN matches the targeted entry.

Server Console Java-based application that allows you to perform administrative management of your Directory Server from a GUI.

server daemon The server daemon is a process that, once running, listens for and accepts requests from clients.

server service The server service is a process on Windows NT that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.

server root A directory on the server machine dedicated to holding the server program and configuration, maintenance, and information files.

Server Selector Interface that allows you select and configure servers using a browser.

service A background process on a Windows NT machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SIE Server Instance Entry.

Simple Network Management Protocol *See SNMP.*

single-master replication The most basic replication scenario in which two servers each hold a copy of the same read-write replicas to consumer servers. In a single-master replication scenario, the supplier server maintains a change log.

SIR *See supplier-initiated replication.*

slapd LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication. See also *ns-slapd*.

SNMP Simple Network Management Protocol. Used to monitor and manage application processes running on the servers, by exchanging data about network activity.

SNMP master agent Software that exchanges information between the various subagents and the NMS.

SNMP subagent Software that gathers information about the managed device and passes the information to the master agent.

SSL Secure Sockets Layer. A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP.

standard index Indexes that are maintained by default.

sub suffix A branch underneath a root suffix.

subagent *See SNMP subagent.*

substring index Allows for efficient searching against substrings within entries. Substring indexes are limited to a minimum of two characters for each entry.

suffix The name of the entry at the top of the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.

superuser The most privileged user available on Unix machines (also called root). The superuser has complete access privileges to all files on the machine.

supplier Server containing the master copy of directory trees or subtrees that are replicated to consumer servers.

supplier server In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

supplier-initiated replication Replication configuration where supplier servers replicate directory data to consumer servers.

symmetric encryption Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.

system index Cannot be deleted or modified as it is essential to Directory Server operations.

target In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entry The entries within the scope of a CoS.

TCP/IP Transmission Control Protocol / Internet Protocol. The main network protocol for the Internet and for enterprise (company) networks.

template entry *See CoS template entry*.

time / date format Indicates the customary formatting for times and dates in a specific region.

TLS Transport Layer Security. The new standard for secure socket layers, a public key based protocol.

topology The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security *See TLS*.

uid A unique number associated with each user on a Unix system.

URL Uniform Resource Locator. The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is `[protocol]://[machine:port]/[document]`. The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

virtual list view index Otherwise known as a browsing index, speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branchpoint in the directory tree to improve display performance.

X.500 standard The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementation.

Index