Sun Java™ System

Directory Server 5.2 Deployment Planning Guide

2004Q2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-5218-10

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registeredtrademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, JDK, Java Naming and Directory Interface, JavaMail, JavaHelp, J2SE, iPlanet, the Duke logo, the Java Coffee Cup logo, the Solaris logo, the SunTone Certified logo and the Sun ONE logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S.and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

Legato and the Legato logo are registered trademarks, and Legato NetWorker, are trademarks or registered trademarks of Legato Systems, Inc. The Netscape Communications Corp logo is a trademark or registered trademark of Netscape Communications Corporation.

The OPEN LOOK and Sun(TM) Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

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.

Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse http://www.sun.com/patents et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays.

CE PRODUIT CONTIENT DES INFORMATIONS CONFIDENTIELLES ET DES SECRETS COMMERCIAUX DE SUN MICROSYSTEMS, INC. SON UTILISATION, SA DIVULGATION ET SA REPRODUCTION SONT INTERDITES SANS L AUTORISATION EXPRESSE, ECRITE ET PREALABLE DE SUN MICROSYSTEMS, INC.

Cette distribution peut comprendre des composants développés par des tierces parties.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, JDK, Java Naming and Directory Interface, JavaMail, JavaHelp, J2SE, iPlanet, le logo Duke, le logo Java Coffee Cup, le logo Solaris, le logo SunTone Certified et le logo Sun[tm] ONE sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

Legato, le logo Legato, et Legato NetWorker sont des marques de fabrique ou des marques déposées de Legato Systems, Inc. Le logo Netscape Communications Corp est une marque de fabrique ou une marque déposée de Netscape Communications Corporation.

L'interface d'utilisation graphique OPEN LOOK et Sun(TM) a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une license non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, seconforment aux licences écrites de Sun.

Les produits qui font l'objet de ce manuel d'entretien et les informations qu'il contient sont regis par la legislation americaine en matiere de controle des exportations et peuvent etre soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucleaires, des missiles, des armes biologiques et chimiques ou du nucleaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou reexportations vers des pays sous embargo des Etats-Unis, ou vers des entites figurant sur les listes d'exclusion d'exportation americaines, y compris, mais de maniere non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une facon directe ou indirecte, aux exportations des produits ou des services qui sont regi par la legislation americaine en matiere de controle des exportations et la liste de ressortissants specifiquement designes, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Contents

Preface

The *Directory Server Deployment Planning Guide* contains the information you need in order to plan your directory deployment, and to make up front decisions on issues such as data types, access control, replication, and sizing.

This preface contains the following sections:

- Who Should Read This Guide
- How This Guide Is Organized
- Using the Documentation
- Conventions
- Resources and Tools on the Web
- How to Report Problems
- Sun Welcomes Your Comments

Before performing any of the tasks described in this guide, read the *Directory Server Release Notes*.

Who Should Read This Guide

This guide is intended for directory decision-makers, solution designers, and administrators.

The author of this guide assumes you are familiar with the following:

- Specifications for LDAP and related protocols
- Clustering model (if you are using Directory Server with Sun Cluster software)
- Internet and World Wide Web technologies

How This Guide Is Organized

This guide is divided into these chapters:

- Directory Server Overview

  Provides an overview of the Directory Server architecture, and describes at a high level the design and deployment process.

- Planning and Accessing Directory Data

  Describes the issues and strategies behind planning and accessing directory data.

- Directory Server Schema

  Describes the schema design process, including mapping data to the default schema, customizing the schema, and maintaining data consistency.

- The Directory Information Tree

  Outlines the steps for designing a directory tree, including grouping directory entries and managing attributes.

- Distribution, Chaining, and Referrals

  Describes how to plan your directory topology, including the distribution of data, referrals, and chaining.

- Understanding Replication

  Provides an overview of replication concepts and discusses the use of replication in your deployment.

- Access Control, Authentication, and Encryption

Describes how to analyze your security needs and explains how to design your directory to meet those needs.

- Directory Server Monitoring

  Examines what an effective monitoring strategy should include, and presents the monitoring features provided with Directory Server.

- Reference Architectures and Topologies

  Provides architectural strategies that outline the role of the physical location of your data, how and where this data is replicated, what you can do to minimize failures, and how to react when failures do occur.

- System Sizing

  Suggests ways of estimating disk and memory requirements for a Directory Server instance. It also touches on network and SSL accelerator hardware requirements.

Using the Documentation

The Directory Server manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML) formats. Both formats are readable by assistive technologies for users with disabilities. The Sun™ documentation web site can be accessed here:

`http://docs.sun.com`

The Directory Server documentation set can be accessed here:

`http://docs.sun.com/coll/DirectoryServer_04q2`

Table 1 briefly describes each document in the set. The left column provides the name and Web location of each document. The right column describes the general contents of the document.

Table 1 Directory Server Documentation

Document	**Contents**
Directory Server Release Notes `http://docs.sun.com/doc/817-5216`	Contains the latest information about Directory Server, including known problems.
Directory Server Technical Overview `http://docs.sun.com/doc/817-5217`	Provides a quick look at many key features of Directory Server.

Table 1 Directory Server Documentation *(Continued)*

Document	Contents
Directory Server Deployment Planning Guide http://docs.sun.com/doc/817-5218	Explains how to plan directory topology, data structure, security, and monitoring, and discusses example deployments.
Directory Server Installation and Migration Guide http://docs.sun.com/doc/817-5219	Covers update, upgrade, and data migration procedures for moving to the latest version of Directory Server.
Directory Server Performance Tuning Guide http://docs.sun.com/doc/817-5220	Provides tips and explanations you can use to optimize Directory Server performance.
Directory Server Administration Guide http://docs.sun.com/doc/817-5221	Gives the procedures for using the console and command-line to manage your directory contents and configure every feature of Directory Server.
Directory Server Administration Reference http://docs.sun.com/doc/817-5235	Details the Directory Server configuration parameters, commands, files, error messages, and schema.
Directory Server Plug-In Developer's Guide http://docs.sun.com/doc/817-5222	Demonstrates how to develop Directory Server plug-ins.
Directory Server Plug-In Developer's Reference http://docs.sun.com/doc/817-5223	Details the data structures and functions of the Directory Server plug-in API.

Conventions

Table 2 describes the typeface conventions used in this guide.

Table 2 Typeface Conventions

Typeface	Meaning	Examples
`AaBbCc123` (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, on-screen computer output, sample code.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123`
(Monospace bold)
	What you type, as contrasted with on-screen computer output.	`%` **`su`**
		`Password:`

Table 2 Typeface Conventions *(Continued)*

Typeface	Meaning	Examples
AaBbCc123 (Italic)	Book titles.	Read Chapter 6 in the *Developer's Guide*.
	New words or terms.	These are called *class* options.
	Words to be emphasized.	You *must* be superuser to do this.
	Command-line variables to be replaced by real names or values.	The file is located in the *ServerRoot* directory.

Table 3 describes placeholder conventions used in this guide.

Table 3 Placeholder Conventions

Item	Meaning	Examples
install-dir	Placeholder for the directory prefix under which software binaries reside after installation.	The default *install-dir* on Solaris systems is `/`. The default *install-dir* on Red Hat systems is `/opt/sun`.
ServerRoot	Placeholder for the directory where server instances and data reside. You can manage each server under a *ServerRoot* remotely through your client-side Server Console. The Server Console uses the server-side Administration Server to perform tasks that must execute directly on the server-side system.	The default *ServerRoot* directory is `/var/opt/mps/serverroot.`
slapd-*serverID*	Placeholder for the directory where a specific server instance and its associated data resides by default.	The default *serverID* is the host name.

Table 4 describes the symbol conventions used in this book.

Table 4 Symbol Conventions

Symbol	Meaning	Notation	Example
`[ ]`	Contain optional command options.	`O[`*n*`]`	`-O4, -O`
`{ }`	Contain a set of choices for a required command option.	`d{y|n}`	`-dy`
|	Separates command option choices.		

Table 4 Symbol Conventions *(Continued)*

Symbol	Meaning	Notation	Example
+	Joins simultaneous keystrokes in keyboard shortcuts that are used in a graphical user interface.		Ctrl+A
-	Joins consecutive keystrokes in keyboard shortcuts that are used in a graphical user interface.		Esc-S
>	Indicates menu selection in a graphical user interface.		File > New File > New > Templates

Table 5 describes the shell prompt conventions used in this book.

Table 5 Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Input and output of Directory Server commands are usually expressed using the LDAP Data Interchange Format (LDIF) [RFC 2849] . Lines are wrapped for readability.

Resources and Tools on the Web

The following location contains information about Java Enterprise System and its component products such as Directory Server:

`http://wwws.sun.com/software/learnabout/enterprisesystem/index.html`

Some supported platforms provide native tools for accessing Directory Server. For more tools useful when testing and maintaining LDAP directory servers, download the Sun Java System Directory Server Resource Kit (DSRK). This software is available at the following location:

`http://wwws.sun.com/software/download/`

Installation instructions and reference documentation for the DSRK tools is available in the *Directory Server Resource Kit Tools Reference*.

For developing directory client applications, you may also download the Sun Java System Directory SDK for C and the Sun Java System Directory SDK for Java from the same location.

Additionally, Java Naming and Directory Interface (JNDI) technology supports accessing Directory Server using LDAP and DSML v2 from Java applications. Information about JNDI is available from:

`http://java.sun.com/products/jndi/`

The JNDI Tutorial contains detailed descriptions and examples of how to use JNDI. It is available at:

`http://java.sun.com/products/jndi/tutorial/`

Third-party URLs are included in this document to provide additional, related information.

NOTE Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

How to Report Problems

If you have problems with Directory Server, contact Sun customer support using one of the following mechanisms:

- Sun Software Support services online at

  http://www.sun.com/service/sunone/software

  This site has links to the Online Support Center and ProductTracker, as well as to maintenance programs and support contact numbers.

- The SunSolve support website at

  http://sunsolve.sun.com

  This site includes patches, support documents, security information, and the Sun System Handbook.

- The telephone dispatch number associated with your maintenance contract

So that we can best assist you in resolving problems, please have the following information available when you contact support:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Use the web-based form to provide feedback to Sun:

http://www.sun.com/hwdocs/feedback/

Please provide the full document title and part number in the appropriate fields. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the part number of this Deployment Planning Guide is 817-5218-10.

Chapter 1

Directory Server Overview

Directory Server provides a centralized directory service for your intranet, network, and extranet information. It 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.

An introduction to basic LDAP and directory concepts is provided in the *Directory Server Technical Overview*. This chapter provides an overview of the server architecture, and describes at a high level the design and deployment process, including issues to be taken into account when planning a Directory Server installation. It is divided into the following sections:

- Server Architecture Overview
- Directory Design Overview
- Directory Deployment Overview

Server Architecture Overview

Any Directory Server deployment includes the following elements:

- Directory Server
- Administration Server
- Sun Java System Server Console

Each of these elements plays a separate role in the deployment.

Directory Server stores the server and application configuration settings, as well as the user information used by other servers in the enterprise. Typically, application and server configuration information is stored in one *suffix* of Directory Server while user and group entries are stored in another suffix. (A suffix refers to the name of the entry in the directory tree, below which data is stored.)

The *configuration directory* or Configuration Directory Server (CDS) stores information about how Directory Server itself is configured. This directory is generally installed first, and every subsequent server registers with it. A single configuration directory provides for centralized administration of all servers.

The *user directory* stores entries for users and groups who access directory services. The user directory is generally unique to the network domain, and other servers access it for user and group information. A single user directory provides for centralized administration of users and groups.

For small deployments, it is possible to install configuration, user, and other directories on the same directory instance. For larger deployments, consider placing the configuration and user directories on separate servers.

Server Console is the front-end management application for all Sun Java System servers. It finds all servers and applications registered in the configuration directory, displays them in a graphical interface, and lets you manage and configure them.

When you log in to Server Console, it connects to an instance of Administration Server using the Hypertext Transfer Protocol (HTTP.) Administration Server manages requests for all Sun Java System products installed in a single root folder.

For more information on this architecture, see Chapter 1, “Remote Server Administration Overview” in the *Administration Server Administration Guide*.

Directory Design Overview

The directory design phase involves gathering data about your directory requirements, such as environment and data sources, users, and the applications that will use the directory.

The flexibility of Directory Server enables you to rework your design to meet unexpected or changing requirements, even after deployment. However, the more modifications you can avoid through good design, the better.

The design process can be broken into the following steps:

- Planning the Installation

- Planning Data and Data Access
- Designing the Schema
- Designing the Directory Tree
- Designing the Topology
- Designing the Replication Process
- Designing a Secure Directory
- Planning a Monitoring Strategy

Planning the Installation

Before installing Directory Server, ensure that you have taken the following into consideration:

1. If the deployment involves centralized administration of server configuration, users, and groups for multiple directory installations, determine the appropriate configuration and user directory locations. Refer to the *Administration Server Administration Guide* for details on appropriate location of configuration, user, and group data.

2. Restrict physical access to the host system. Although Directory Server includes a number of security features, your directory security is compromised if physical access to the host system is not controlled.

3. Ensure the host system uses a static IP address.

4. If the Directory Server instance is not itself providing a naming service for the network or if the deployment involves remote administration of Directory Server, ensure a naming service and the domain name for the host are properly configured.

- Select the port number you will use for each Directory Server instance at design time, and do not change that port number once your Directory Server is in production, if possible. Changing the port number via the console at a later stage does not make the necessary changes to certain scripts and requires that these scripts be modified manually. See the *Directory Server Release Notes* for more information.

Planning Data and Data Access

Your directory will contain data, such as user names, telephone numbers, and group details. Refer to Chapter 2, “Planning and Accessing Directory Data,” for information on analyzing the various sources of data in your organization and understanding their relationship with one another. This chapter describes the types of data appropriate for storage in a directory, the ways in which this data can be accessed, and other tasks you must perform when designing the contents of a directory.

Designing the Schema

Directory Server is designed to support directory-enabled applications. These applications have specific requirements of the data stored in the directory. The schema determines the characteristics of the stored data. Chapter 3, “Directory Server Schema,” introduces the standard schema shipped with 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 this data. This is the purpose of the directory tree. Chapter 4, “The Directory Information Tree,” introduces the directory tree, and guides you through the design of your data hierarchy. It also describes the mechanisms used to optimize entry grouping and attribute management, and provides sample directory tree designs.

Designing the Topology

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

Designing the Replication Process

With replication, multiple Directory Servers maintain the same directory data to increase read performance and provide fault tolerance. Chapter 6, "Understanding Replication," 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

It is essential that you plan how to protect the data in your directory and design the other aspects of your service to meet the security requirements of your users and applications. Chapter 7, "Access Control, Authentication, and Encryption," 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 directory data.

Planning a Monitoring Strategy

A well-designed monitoring strategy will enable you to evaluate the success of your directory deployment and to follow day-to-day directory activities. Chapter 8, "Directory Server Monitoring," discusses how to monitor your directory using SNMP, Directory Server Console, the log files, database monitoring, and the replication monitoring tools provided with Directory Server.

Directory Deployment Overview

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 the service can handle your user load. If the service is not adequate, adjust your design and pilot it again. Adjust your pilot design until you have a robust service that 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 administering and maintaining your directory, refer to the *Directory Server Administration Guide*.

Chapter 2

Planning and Accessing Directory Data

The data stored in your directory may include user names, e-mail addresses, telephone numbers, and information about groups users belong to, 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. Directory Server enables you to access directory data either via LDAP or DSML, extending the types of applications that can interact directly with the data.

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

- Introduction to Directory Data
- Defining Your Data Needs
- Performing a Site Survey
- Accessing Directory Data With DSML Over HTTP/SOAP

Introduction to Directory Data

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

- It is read more often than written.

  Because the directory is tuned for read operations, write operations slow down server performance.

- It is expressible in attribute-value 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 is 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 accesses the information. However, if the application is capable of reading preferences from the directory and users interact with the application according to their preferences from different sites, it is useful to include the preference information in the directory.

What Your Directory Might Include

Examples of data you can store in your directory are:

- Contact information, such as telephone numbers, physical addresses, and e-mail 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.

Apart from server administration data, you may want to store the following types of information in your directory:

- 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 well suited to managing large quantities of data that client applications read and occasionally write, but it is not designed to handle large 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 Directory Server works best for read operations, you should avoid placing rapidly changing information in the directory. Reducing the number of write operations improves overall search performance.

Defining Your Data Needs

When you design your directory data, try to 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 will influence how the data is structured and distributed.

As you plan your deployment, consider the following:

- 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 or DSML-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 to your current customers. Maybe future customers will want to use your directory to store JPEG images. 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 of discovering and characterizing the contents of a 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 then in more detail:

- Identify the applications that use the directory.

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

- Identify how the applications will access the directory.

  Determine which mode of access - using LDAP or DSML over HTTP/SOAP - your applications will use.

- Identify data sources.

  Survey your enterprise and identify sources of data (such as NT or Netware directories, PBX systems, human resources databases, e-mail systems, and so forth).

- Characterize the data the directory must contain.

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

- Determine the level of service you must provide.

  Decide how available the directory data must be to client applications and design your architecture accordingly. How available your directory must 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, “Understanding Replication.” For more information on chaining, refer to Chapter 5, “Distribution, Chaining, and Referrals.”

- 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 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 and reduces administrative overhead.

- Document the 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 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.

- Repeating the site survey.

  If your enterprise has more than one office you should repeat the survey to ensure that each office has been taken into account. It is advisable to set up site survey teams in each location, who feed their results back into a central site survey team (with representatives from each location).

Identifying Client Applications

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

- Directory browser applications, such as white pages. These kinds of applications generally access information such as e-mail addresses, telephone numbers, and employee names.
- Messaging applications, especially e-mail servers. All e-mail servers require e-mail addresses, user names, and some routing information. Others 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 details, and job titles.
- Security, web portal, or personalization applications. These kinds of applications access profile information.

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, e-mail address, phone number, user ID, password, department number, manager, mail stop
Web server	People, groups	User ID, password, group name, group members, group owner
Calendar server	People, meeting rooms	Name, user ID, cube number, conference room name
Web portal	People, groups	Name, User ID, password, group name, group members

When you have identified the applications and information used by each application, you may 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.

The data maintained in your directory, and when it starts being maintained, is affected by:

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

Identifying Data Sources

To identify the data to be included in your directory, perform a survey of existing data sources. 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, Novell Netware, UNIX NIS), e-mail systems, security systems, PBX (telephone switching) systems, and human resources applications.

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

  Centralized data management may require new tools and new processes. Issues may arise when centralization requires increasing staff in some organizations and 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
E-mail system	People, Groups	Name, e-mail address, user ID, password, e-mail preferences
Facilities system	Facilities	Building names, floor names, cube numbers, access codes

Characterizing Directory Data

The data you identify can be characterized as follows:

- Format
- Size
- Number of occurrences in various applications

- Data owner
- Relationship to other directory data

Study each piece of data you plan to include in your directory to determine what characteristics it shares with other pieces of data. This helps save time during the schema design stage, described in Chapter 3, “Directory Server 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
E-mail address	Text	Many characters	IS department	User’s entry

Determining Directory Availability Requirements

The level of service you provide, in terms of availability, depends on the expectations of those who rely on directory-enabled applications. To determine the level of service an application expects, first determine when and how the application is used.

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

Considering a Data Master Server

The data master is the server that is the primary source of data. If you have more than one data center (physical site) you need to decide which server will be the data master, and which servers receives updates from this data master.

Data Mastering for Replication

If you use replication, decide which server will be the master source of your data. Directory Server supports multi-master configurations, in which more than one server can be a master for the same data. For more information about replication and multi-master replication, see Chapter 6, “Understanding Replication.”

In the simplest case, put a master source of all your data on two Directory Servers and then replicate that data to one or more consumer servers. Having two master servers provides failover in the event that a server goes offline. 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 that 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 Sun Java System Meta Directory.

  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 Sun Java System sales representative for more information about Meta Directory.

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

  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 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.

Suppose you want to manage an employee’s home telephone number. Both the LDAP directory and a human resources (HR) database store this information. The HR database is LDAP enabled, so you can write an application that automatically transfers data from the LDAP directory to the HR database, and vice versa. However, if you attempt to master changes to the telephone number in both the LDAP directory and the HR database, the last place where the telephone number was changed overwrites the information in the other database. This is fine if the last application to write the data had the correct information. But if that information was out of date (because the HR data was reloaded from a backup, for example), 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 phase, decide who can write data to the directory. Common strategies for determining data ownership include the following:

- Allow read-only access to the directory for everyone except a small group of directory content managers.
- Allow individual users to manage strategic subsets of information themselves.

  These subsets 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 makes your organization 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 (social security number), and home phone numbers and address.

For more information about roles and grouping entries, refer to Chapter 4, "The Directory Information Tree."

As you determine who can write to the data, you may find that multiple individuals require write access to the same information. For example, you will want an information systems 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.

For information on setting access control for your directory, see Chapter 7, "Access Control, Authentication, and Encryption."

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 stored in your directory, 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, e-mail addresses, and business telephone numbers. However, anonymous access gives anyone with access to the directory access to the common information. You should therefore 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 (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 control, refer to "Designing Access Control" on page 161.

- 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, "The Directory Information Tree." For information about access controls, see Chapter 7, "Access Control, Authentication, and Encryption."

As you make these decisions for each piece of directory data, you define a security policy for your directory. Your decisions depend on the nature of your site and the kinds of security already available. For example, if your site has a firewall or no direct access to the Internet, you may feel more free 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 that represent them. Check with your organization's legal department to ensure that your directory deployment follows the 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, "Access Control, Authentication, and Encryption."

Documenting Your Site Survey

Because of the complexity of data design, it is advisable that you 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.

A basic data tracking example is provided in Table 2-4. This table identifies data ownership and data access for each piece of data identified by the site survey.

Table 2-4 Data Tracking Table Example for Site Survey Documentation Purposes

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

The row representing the employee name data contains the following:

- Owner

  Human resources owns this information and is therefore 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 HR group can add, change, and delete employee names.

- IS Writable

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

Repeating the Site Survey

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.

Accessing Directory Data With DSML Over HTTP/SOAP

Previous versions of Directory Server enabled you to access directory data using the Lightweight Directory Access Protocol (LDAP). Directory Server 5.2 also enables you to access directory data using Directory Service Markup Language version 2 (DSMLv2) over HTTP/SOAP.

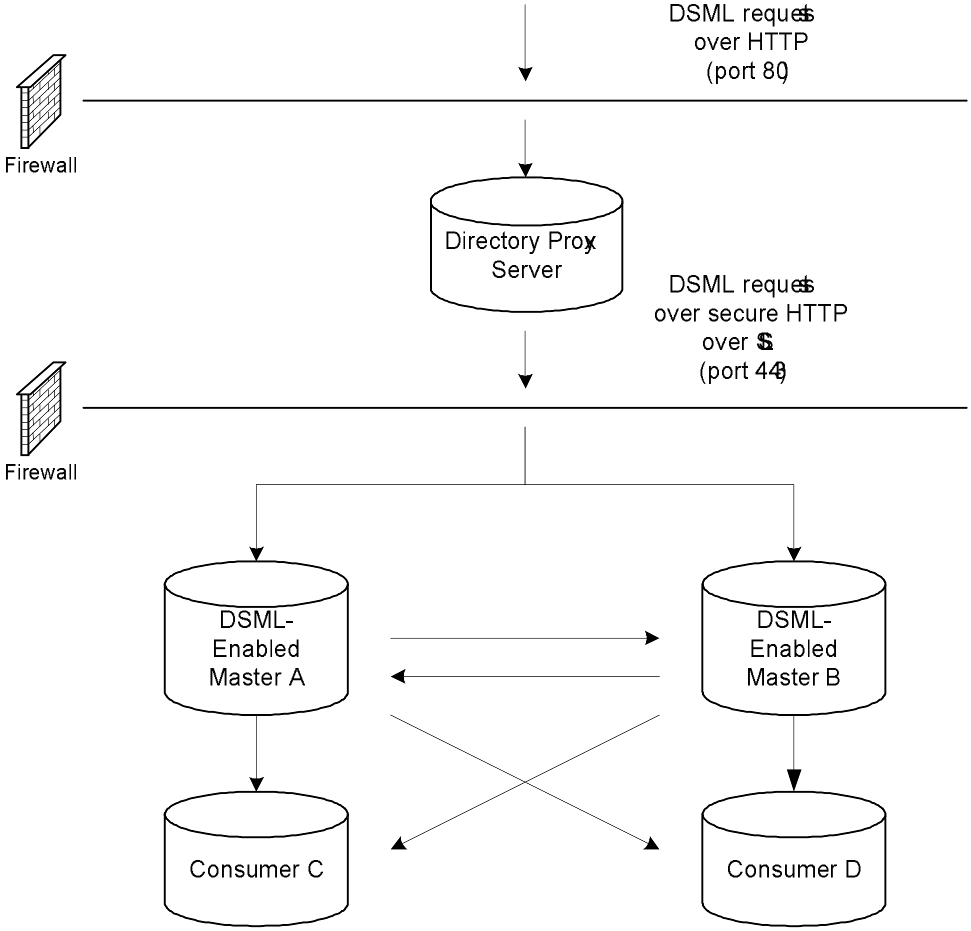
DSMLv2 is a markup language, that is, a vocabulary and schema that enables users to describe the structure and content of directory services data operations in an eXtensible Markup Language (XML) document. DSMLv2 standardizes the way directory services information is represented in XML. Directory Server supports the use of DSMLv2 over the Hypertext Transfer Protocol (HTTP/1.1) and uses the Simple Object Access Protocol (SOAP) version 1.1 as a programming protocol to transport the DSML content.

For information on configuring the DSML frontend and on accessing and searching data using DSMLv2 over HTTP/SOAP, see “Configuring DSML,” in Chapter 1 of the *Directory Server Administration Guide*.

DSMLv2 Over HTTP/SOAP Deployment

The following sample deployment using DSML-enabled Directory Servers and Sun Java System Web Proxy Server, enables non-LDAP clients to interact with directory data.

Figure 2-1 Sample DSML-Enabled Directory Deployment



In this sample deployment, update requests in DSML arriving from non-LDAP client applications cross a firewall over HTTP port 80 and enter a demilitarized zone (DMZ.) From there Directory Proxy Server configured as a reverse proxy server enforces the use of secure HTTP over port 443 for the requests to cross a second firewall and enter the intranet domain. The requests are then processed by the two master replicas on Master A and Master B, before being replicated to the non-DSML enabled Consumers C and D.

This deployment enables non-LDAP applications to perform directory operations. If the client requests are solely lookup requests, it is irrelevant whether the DSML-enabled Directory Servers hold read-only or read-write copies of the data, because both would be able to process the lookup requests. However, if a non-LDAP client issues modification requests, it is important for the DSML-enabled Directory Servers to hold read-write copies of the data. The default behavior for a consumer receiving a modification request is to return a referral with a list of LDAP URLs of the possible masters that could satisfy the request. Returning an LDAP URL over HTTP to a non-LDAP client application would not fulfill the objective of keeping client/directory traffic LDAP-free, which is why read-write copies are preferable. The deployment depicted in Figure 2-1, holds read-write copies of the data on the DSML-enabled Directory Servers Master A and Master B. These masters process modification requests and then replicate the data to the non-DSML enabled Consumers C and D.

The DSML front end constitutes a restricted HTTP server. It accepts only DSML HTTP post operations, and rejects requests that do not conform to the SOAP/DSML specification. Therefore, the threat is less extensive than for other types of HTTP web server. Nonetheless, you should take into account the following security considerations when including DSML-enabled Directory Servers in your deployment:

- Protect DSML-enabled Directory Servers by implementing a firewall.
- Use secure HTTP over SSL on port 443 or implement a web proxy server solution, if you prefer not to impose the use of HTTP over SSL on your clients.

Chapter 3

Directory Server Schema

The site survey conducted in Chapter 2 provided information about the data you plan to store in your directory. Next, you must decide how to represent this data. The directory schema describes the types of data that can be stored in a directory. During schema design, each data element is mapped to an LDAP attribute, and related elements are gathered into LDAP object classes. Well-designed schema helps maintain data integrity.

This chapter describes how to design schema, and includes the following sections:

- Directory Server Schema
- Schema Design Process
- Mapping Your Data to the Default Schema
- Customizing the Schema
- Maintaining Data Consistency
- Other Schema Resources

For more information about the object classes and attributes found in Directory Server, in addition to the schema files and directory configuration attributes, refer to the *Directory Server Administration Reference*. For information on replicating schema between servers, refer to "Schema Replication" on page 138.

Directory Server Schema

The directory schema maintains data integrity 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 the standard RFC LDAP schema, additional application-specific schema to support the features of the server, and Directory Server-specific schema extensions. While this schema meets most directory requirements, 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" on page 35 for information on extending the schema.

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 retrieve the schema programmatically and to adapt their behavior based on it. The global set of schema for Directory Server can be found in the entry `cn=schema`.

The Directory Server schema supports not only the core LDAPv3 schema in RFC 2256, but many other popular product schemas as well. In addition to this, Directory Server 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 Sun for Directory Server's use, the X-ORIGIN field contains the value `Sun ONE 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 (objectclass $ 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`), and a description of the class (`Standard Person Object Class`), then lists the required attributes (`objectclass`, `sn`, and `cn`) and the allowed attributes (`description`, `seealso`, `telephoneNumber`, and `userPassword`).

Like all Directory Server 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.

Schema Design Process

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.

Where possible, it is best to use the 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 that it has been reviewed and agreed to by a large number of directory users.

Mapping Your Data to the Default Schema

The data you identified during your site survey, (see "Performing a Site Survey" on page 18) must be mapped to the default directory schema. This section describes how to view the default schema and provides a method for mapping your data to the appropriate schema elements.

If you find elements in your schema that do not match the default schema, you may need to create custom object classes and attributes. Refer to "Customizing the Schema" on page 35 for more information.

Viewing the Default Directory Schema

The schema provided with Directory Server is described in a set of files stored in the following directory:

ServerRoot`/slapd-`*serverID*`/config/schema`

This directory contains all of the common schema for the Sun Java System products. The LDAPv3 standard user and organization schema is located in the `00core.ldif` file. The configuration schema used by earlier versions of the directory is located in the `50ns-directory.ldif` file.

NOTE You should never modify the files in this directory while the server is running.

Also, any changes made manually will *not* be replicated until other changes are made using either LDAP or Directory Server Console.

Matching Data to Schema Elements

The data identified in your site survey must now 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 must be noted in your 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 must be considered as different objects in your schema.

- Select a similar object class from the default schema.

  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 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 35 for more information.

The following table maps directory schema elements to the data identified during the site survey:

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 Table 3-1, the employee name describes a person. The default directory schema contains 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, there is not 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 `organizationalPerson` object class, which in turn inherits from the `person` object class. The `inetOrgPerson` object's allowed attributes include 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 is too limited for your directory needs. Directory Server Console assists in managing the schema definition. For more information, refer to Chapter 9, "Extending the Directory Schema" in the *Directory Server Administration Guide*.

Keep the following rules in mind when customizing schema:

- Reuse existing schema elements whenever possible. For a complete list of the existing schema elements, refer to "Object Class Reference" and "Attribute Reference" in the *Directory Server Administration Reference*.
- 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.

NOTE When customizing the schema, do not modify, delete, or replace any existing definitions of attributes or object classes in the standard schema. Doing so can lead to compatibility problems with other directories or other LDAP client applications.

Custom object classes and attributes are defined in the following file:

ServerRoot/`slapd-`*serverID*`/config/schema/99user.ldif`

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

- When to Extend Your Schema
- Obtaining and Assigning Object Identifiers
- Naming Attributes and Object Classes
- Strategies for Defining New Object Classes
- Strategies for Defining New Attributes
- Deleting Schema Elements
- Creating Custom Schema Files - Best Practices and Pitfalls

When to Extend Your Schema

While the object classes and attributes supplied with 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.

Obtaining 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. Obtaining 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 used 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 Attributes and Object Classes

When creating names for new attributes 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 custom schema elements and existing schema elements by including a unique prefix on custom elements. For example, Example.com Corporation might add the prefix `Example` before each of their custom schema elements. They might add a special object class called `ExamplePerson` to identify Example.com employees in their directory.

Strategies for Defining New Object Classes

Add new object classes when the existing object classes do not support all of the information you need to store in a directory entry. There are two ways to 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 object class.

You may find it easiest to mix the two methods.

Suppose your site wants to create the attributes `ExampleDepartmentNumber`, and `ExampleEmergencyPhoneNumber`. You can create several object classes that allow some subset of these attributes. You might create an object class called `ExamplePerson` and have it allow `ExampleDepartmentNumber` and `ExampleEmergencyPhoneNumber`. The parent of `ExamplePerson` would be inetOrgPerson. You might then create an object class called ExampleOrganization and have it also allow ExampleDepartmentNumber and ExampleEmergencyPhoneNumber. The parent of ExampleOrganization would be the organization object class.

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

```
objectclasses: ( 1.3.6.1.4.1.42.2.27.999.1.2.3 NAME 'ExamplePerson'
 DESC 'Example Person Object Class' SUP inetorgPerson STRUCTURAL MAY
 (ExampleDepartmentNumber $ ExampleEmergencyPhoneNumber) )
```

```
objectclasses: ( 1.3.6.1.4.1.42.2.27.999.1.2.4 NAME
 'ExampleOrganization' DESC 'Example Organization Object Class' SUP
 organization STRUCTURAL MAY (ExampleDepartmentNumber
 $ ExampleEmergencyPhoneNumber) )
```

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: (1.3.6.1.4.1.42.2.27.999.1.2.5 NAME 'ExampleEntry'
 DESC  'Example Auxiliary Object Class' SUP top  AUXILIARY MAY
 (ExampleDepartmentNumber $ ExampleEmergencyPhoneNumber) )
```

The new ExampleEntry 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 examples is based on the Sun Java System OID prefix and must not be used in the deployed product. To create your own new object classes, you must obtain your own OID. For more information, refer to "Obtaining and Assigning Object Identifiers" on page 36.

Consider the following when deciding how to implement new object classes:

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

  Generally, the number of elements remains small and needs 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 STRUCTURAL object classes require 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. You may find this to be either helpful or cumbersome.

- Single AUXILIARY 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.

- Try to design object classes which relate to real objects and group elements that constitute sensible groupings.
- Avoid required attributes for new object classes.

  Requiring attributes can make your schema inflexible. When you 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 which object class(es) it inherits.

Strategies for Defining New Attributes

Add new attributes when the existing attributes 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.

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 birth dates in your directory, no attribute exists within the standard Directory Server schema. You can create a new attribute called dateOfBirth and allow this attribute to be used on entries representing people by defining a new auxiliary class 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. If you delete parts of the standard LDAP schema you may run into compatibility problems with future installations of Directory Server and other directory-enabled applications.

If you extend the schema and find that you do not use the new elements, you can delete these unused elements. Before removing schema elements, make sure that no entry in the directory uses them. The easiest way to do this is to run an ldapsearch that returns all entries containing that schema element.

For example, before deleting the object class named `myObjectClass`, you would run the following ldapsearch command:

```
ldapsearch -h host -p port -s base "objectclass=myObjectClass"
```

If you find any such entries, you can delete them or the part that will be removed from the schema. If you remove the schema definition before removing the entries that use that definition, you might not be able to modify the entries afterwards. Schema checks on modified entries will also fail unless you remove the unknown values from the entry.

Creating Custom Schema Files - Best Practices and Pitfalls

You can create custom schema files other than the `99user.ldif` file provided with Directory Server. However, you must bear the following in mind when creating custom schema files, especially when you are using replication:

- When adding new schema elements, all attributes must 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 (the server loads the schema in numerical order first, then in alphabetical order).
- When defining new schema definitions manually it is best practice to add these definitions to the `99user.ldif` file.

  When you update schema elements using LDAP, the new elements are written automatically to the `99user.ldif` file. As a result, any other schema definition changes you may have made in custom schema files may be overwritten. Using only the `99user.ldif` file prevents possible duplications of schema elements and the danger of schema changes being overwritten.
- Because Directory Server loads schema files in alpha-numerical order, (with numbers being loaded first), you should name custom schema files as follows:

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

  where the number is higher than any directory standard schema already defined.

If you name your schema file with a number that is 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.

- Make sure that custom schema filenames are not numerically or alphabetically higher than `99user.ldif` as Directory Server uses the highest sequenced file (numerically, then alphabetically) for its internal schema management.

  If you created a schema file and named it `99zzz.ldif` for example, the next time you updated the schema using LDAP or Directory Server Console, all of the attributes with an X-ORIGIN value of `'user defined'` (usually stored in the `99user.ldif` file) would be written to `99zzz.ldif` instead. The result would be two LDIF files that contain duplicate information, and some information in the `99zzz.ldif` file might be erased.

- As a general rule, you should identify the custom schema elements you are adding with the following two items:

  - `'user defined'` in the `X-ORIGIN` field of custom schema files,
  - a more descriptive label such as `'Example.com Corporation defined'` in the `X-ORIGIN` field, so that the custom schema element is easy to understand for other administrators. For example `X-ORIGIN ('user defined' 'Example.com Corporation defined')`.

  If you are adding schema elements manually and you do not use `'user defined'` in the `X-ORIGIN` field, the schema elements will appear in the read-only section of Directory Server Console and you will not be able to use the console to edit them.

  The `'user defined'` value is added automatically by the server if you add custom schema definitions using LDAP or Directory Server Console. However, if you do not add a more descriptive value in the `X-ORIGIN` field, you may have difficulty understanding what the schema relates to at a later date.

- Propagate any custom schema files manually to all of your servers, because these changes are not replicated automatically.

  When you change the directory schema, the server keeps a time-stamp of when the schema was changed. At the beginning of each replication session the server compares its time-stamp with its consumer's time-stamp and, if necessary, pushes any schema changes. For custom schema files the server maintains only one time-stamp, which is associated with the `99user.ldif` file.

This means that any custom schema file changes or additions you make to files other than 99user.ldif will not be replicated. Therefore, you must propagate custom schema files to all other servers to ensure that all schema information is present throughout the topology.

To propagate custom schema changes you can either:

- Replicate the changes by running the schema_push.pl script, or
- Manually copy these custom schema files to all of your servers.

Both methods require that each server is restarted. If you use the schema_push.pl script to replicate custom schema definitions, you must maintain your schema on one master only. When schema definitions are replicated to a consumer on which they do not already exist, they will be stored in the 99user.ldif file as opposed to the custom schema file in which you defined them. Storing schema elements in the 99user.ldif file of consumers does not create a problem as long as you ensure that you maintain your schema on one master server only.

If you copy your schema files manually, you must remember to copy the files *each* time changes are made. If you do not do this the changes may 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.

- If you do not want custom schema elements to be replicated to other servers in the replication topology:
  - define the schema elements you do not want to replicate in a separate file,
  - do *not* identify these elements as 'user defined' in the X-ORIGIN field,
  - set the nsslapd-schema-repl-useronly attribute to on so that only schema labeled as 'user defined' in the X-ORIGIN field will be replicated.

NOTE You must also set the nsslapd-schema-repl-useronly attribute to on when replicating to a 5.0 or 5.1 Directory Server.

For more information about replicating schema, see "Schema Replication" on page 138.

Maintaining Data Consistency

Maintaining data consistency within Directory Server assists LDAP client applications in locating directory entries. For each type of information you store in the directory, select the required object classes and attributes to support that information, and always use the same ones. If you use schema objects inconsistently, it becomes difficult to locate information efficiently.

You can maintain schema consistency in the following ways:

- Use schema checking to ensure that 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 schema consistency.

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 only checks that the proper attributes are present. It does not verify whether attribute values are in the correct syntax for the attribute. Directory Server 5.2 includes an attribute called `nsslapd-valuecheck` which allows you to check attributes whose values have the DN syntax. However, this attribute is turned off by default, so no attribute values are checked.

By default, the directory enables schema checking. You should not turn schema checking off on a server that is accepting client updates. For information on turning schema checking on and off, refer to “Schema Checking” in Chapter 9 of the *Directory Server Administration Guide*.

With schema checking on, you must take note of the 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, 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`.

When configuring schema checking, bear the following in mind:

- Generally, you replicate all required attributes for each entry as defined in the schema, to avoid schema violations. If you want to filter out certain required attributes using *fractional replication*, you must disable schema checking.
- If schema checking is enabled with fractional replication, you may not be able to initialize the server off line (from an ldif file). This is because the server will not allow you to load the ldif file if required attributes are filtered out.
- Turning schema checking off may improve performance.
- If you have disabled schema checking on a fractional consumer replica, the whole server instance on which that fractional consumer replica resides will not enforce schema. As a result, you should avoid configuring supplier (read-write) replicas on the same server instance.
- Because schema is pushed by suppliers in fractional replication configurations, the schema on the fractional consumer replica will be a copy of the master replica's schema. Therefore, it will not correspond to the fractional replication configuration being applied.

Selecting Consistent Data Formats

LDAP schema allows you to place any data 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 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

Consider the following points for maintaining consistent schema in a replicated environment:

- Do not modify the schema on a consumer server.

  If you modify the schema on a consumer server, it will be more recent than the schema on the master server. When the master sends replication updates to the consumer, you will probably observe a number of replication errors because the schema on the consumer cannot support the new data.

- In a multi-master replication environment, only modify schema on a single master server.

  If you modify the schema on two master servers, the master that was most recently updated will propagate its version of the schema to the consumer. This means that the schema on the consumer will be inconsistent with the schema on the other master.

NOTE In Directory Server 5.2, the schema file `11rfc2307.ldif` has been altered to conform to rfc2307. This file corresponds to `10rfc2307.ldif` (for 5.1 zip installations). If replication is enabled between 5.2 servers and 5.1 servers, the rfc2307 schema *must* be corrected on the 5.1 servers, or replication will not work correctly. Copy the `11rfc2307.ldif` file from the 5.2 instance to the 5.1 instances (and remove the `10rfc2307.ldif` file.)

For more information on schema replication, refer to "Schema Replication" on page 138.

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

The Directory Information Tree

The directory information tree (DIT) provides a way to refer to the data stored in your directory. The types of information stored, the physical nature of your enterprise, the applications that use your directory, and the types of replication you use shape the design of the directory tree. This chapter outlines the steps for designing a directory tree, and includes the following sections:

- Introduction to the Directory Tree
- Designing the Directory Tree
- Grouping Directory Entries and Managing Attributes
- Other Directory Tree Resources

Introduction to the Directory Tree

The directory tree provides a way for directory data to be named and referred to by client applications. The directory tree interacts closely with other design decisions, including how you distribute, replicate, or control access to directory data.

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 the directory
- Simplified directory navigation for users

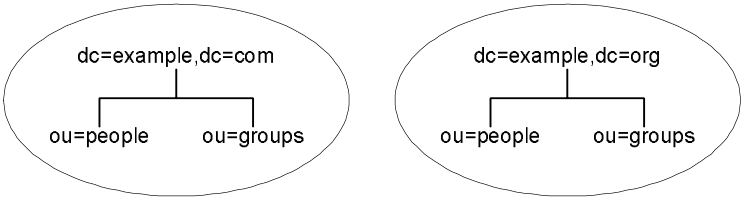
The directory tree structure follows the hierarchical LDAP model. The directory tree organizes data, for example, by group, by people, or by geographical location. It also determines how data is partitioned across multiple servers. Because data can only be partitioned at the suffix level, an appropriate directory tree structure is required to enable you to spread your data across multiple servers.

Directory tree design also has an impact on replication configuration. If you want to replicate only portions of your directory tree, this must be taken into account when you design the directory tree. If you plan to use access controls on branch points, you must also take this into account at design time.

To manage the directory tree, it is defined in terms of suffixes, subsuffixes, and chained suffixes. A *suffix* is a branch or subtree whose entire contents are treated as a unit for administrative tasks. For example, indexing is defined for an entire suffix, an entire suffix may be initialized in a single operation, and a suffix is the unit of replication. Data that you wish to access and manage in the same way should be located in the same suffix. A suffix may be located at the root of the directory tree where it is sometimes called a *root suffix*.

The following figure shows a directory with two root suffixes, each for a separate corporate entity:

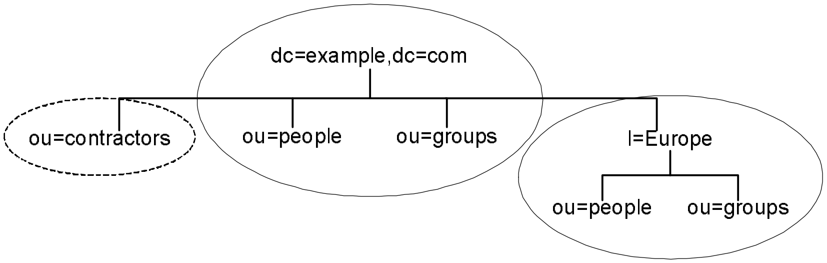
Figure 4-1 Two Root Suffixes in a Single Directory Server



A suffix may also be a branch of another in which case it is called a *subsuffix*. The parent suffix does not include the contents of the subsuffix for administrative operations, and the subsuffix is therefore managed independently of its parent. However, LDAP operation results contain no information about suffixes, and directory clients are unaware of whether entries are part of root suffixes or subsuffixes.

The following figure shows a directory with a single root suffix and multiple subsuffixes for a large corporate entity:

Figure 4-2 One Root Suffix with Multiple Subsuffixes



A suffix corresponds to an individual database within the server. However, databases and their files are now managed internally by the server, and database terminology has been dropped as of Directory Server 5.2.

Chained suffixes create a virtual directory tree by referencing suffixes on other servers. With chained suffixes, Directory Server performs the operation on the remote suffix and returns the result as if it had been performed locally. The location of the data is transparent because the client is unaware that the suffix is chained and that the data is retrieved from a remote server. A root suffix on one server may have sub-suffixes that are chained to another server, thus creating a single tree structure from the client's point of view.

In the special case of cascading chaining, the chained suffix may reference another chained suffix on the remote server, and so on. Each server will forward the operation and eventually return the result to the server handling the client's request.

For more general information about chaining, refer to Chapter 5, "Distribution, Chaining, and Referrals".

Designing the Directory Tree

Directory tree design involves choosing a suffix to contain your data, determining the hierarchical relationship between data entries, and naming the entries in the directory tree hierarchy. The following sections describe the design process in more detail:

- Choosing a Suffix
- Creating Your Directory Tree Structure

- Distinguished Names, Attributes, and Syntax
- Naming Entries

Choosing a Suffix

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

A default Directory Server installation contains multiple suffixes, one for storing data and the others for data needed by internal directory operations (such as configuration information and directory schema). For more information on the default directory suffixes, refer to Chapter 3, "Creating Your Directory Tree" in the *Directory Server Administration Guide*.

Suffix Naming Conventions

All directory entries should be located below a common base entry (the suffix.) Each suffix name should be:

- Globally unique
- Static, so that it rarely changes
- 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, you should choose suffix name that aligns with a DNS or internet domain name. For example, if your enterprise owns the domain name `Example.com`, your suffix would be:

```
dc=example,dc=com
```

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

You can use any attribute to name your suffix. In a hosting environment, however, the suffix should 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 (stateOrProvinceName)`

  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 enables interoperability with subscriber applications. For example, a hosting organization might use these attributes to create the following suffix for one of its clients, Example.com:

```
o=Example.com,st=Washington,c=US
```

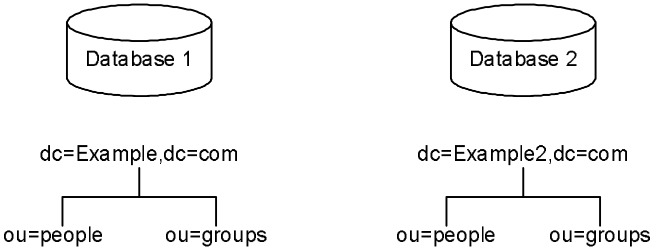
For more information on these attributes, see Table 4-1 on page 53.

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

Working With Multiple Suffixes

Each suffix that you define constitutes a unique directory tree. You can create multiple directory trees, stored in separate databases. For example, you could create separate suffixes for Example.com and Example2.com and store them in separate databases, as illustrated in Figure 4-3:

Figure 4-3 Two Suffixes Stored in Two Different Databases



The databases can be stored on a single server or on multiple servers, depending on resource constraints.

Creating Your Directory Tree Structure

The structure of a directory tree can be flat or hierarchical. As a general rule, your directory tree should be as flat as possible. However, a degree of hierarchy may be required when you partition data across multiple databases, prepare replication, and set access controls.

Designing your directory tree structure includes the following steps:

- Branching Your Directory
- Identifying Branch Point Attributes
- Replication Considerations
- Access Control Considerations

Branching Your Directory

A *branch point* is a point at which you define a new subdivision within the directory tree. When deciding on branch points, avoid potential problematic name changes. The likelihood of a name changing is 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.

The following guidelines will assist you in defining branch points:

- Branch your tree to represent only the largest organizational subdivisions in your enterprise. These branch points should be limited to divisions (Corporate Information Services, Customer Support, Sales and Professional Services, and so forth). Make sure that your divisions are stable; do not perform this kind of branching if your enterprise reorganizes frequently.
- Use functional or generic names rather than actual organizational names. 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, create a single branch point for that function instead of branching based on 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.

The following sections provide sample directory tree structures for an enterprise and hosting environment.

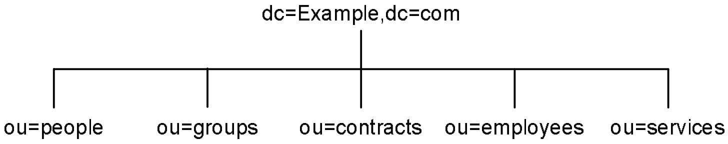
Branching in an Enterprise Environment

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`

Figure 4-4 illustrates a directory tree organized using these objects in a sample enterprise, Example.com.

Figure 4-4 Sample Directory Information Tree Using 5 Branching Points



Try to use only the traditional branch point attributes (shown in Table 4-1). Using traditional attributes increases the likelihood of retaining compatibility with third-party LDAP client applications. In addition, traditional attributes are 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 page50 .

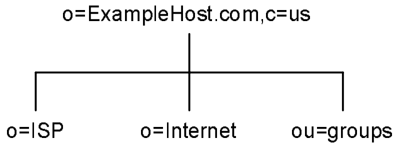
Table 4-1 Traditional DN Branch Point Attributes *(Continued)*

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 50.

Branching in a Hosting Environment

For a hosting environment, create a tree that contains two entries of the `organization(o)` object class and one entry of the `organizationalUnit(ou)` object class beneath the suffix. A sample directory tree for an internet host, ExampleHost.com, is illustrated in Figure 4-5:

Figure 4-5 ISP ExampleHost.com Directory Information Tree



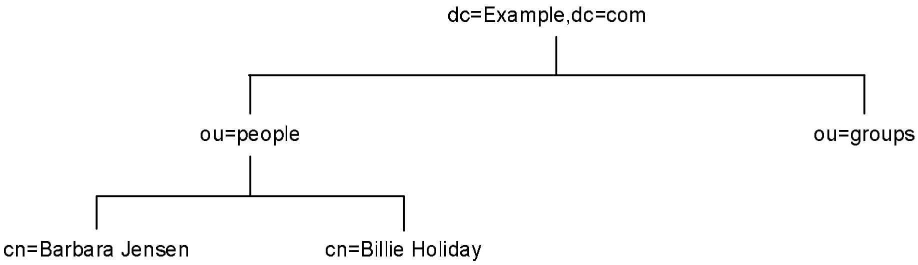
Identifying Branch Point Attributes

As you design your directory tree, you must decide what attributes will be used 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 Example.com Corporation, appears as follows:

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

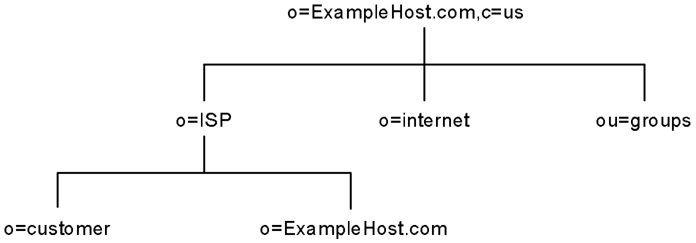
Each attribute-data pair represents a branch point in your directory tree. The directory tree for Example.com Corporation is illustrated in Figure 4-6:

Figure 4-6 Example.com Corporation Directory Information Tree



The directory tree for ExampleHost.com, is illustrated in Figure 4-7:

Figure 4-7 ExampleHost.com Internet Host Directory Information Tree



Beneath the root suffix entry, `o=ExampleHost.com,c=us`, the tree is split into three branches. The ISP branch contains customer data and internal information for ExampleHost.com. The internet branch is the domain tree. The groups branch contains information about the administrative groups.

It is important to be consistent when choosing attributes for your branch points. Some LDAP client applications may be confused if the distinguished name (DN) format is inconsistent across your directory tree. For example, if `l (localityName)` is subordinate to `o (organizationName)` in one part of your directory tree, you must ensure that `l` is subordinate to `o` in all other parts of your directory.

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 in the entry itself.

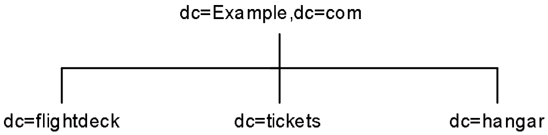
Replication Considerations

When designing your directory tree, consider which entries will be replicated. 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.

In an enterprise environment you can organize your directory tree so that it corresponds to the network names in your enterprise. Network names tend not to 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.

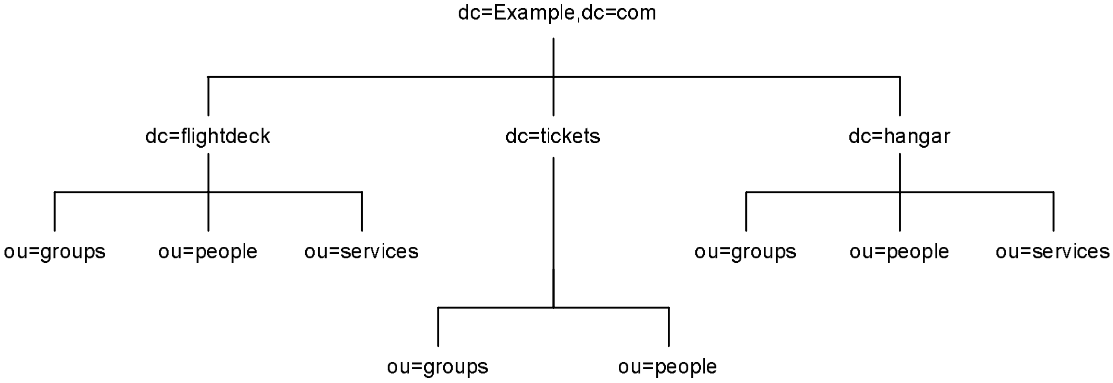
For example, Example.com Corporation has three primary networks known as `flightdeck.Example.com`, `tickets.Example.com`, and `hangar.Example.com`. They initially branch their directory tree as illustrated in Figure 4-8:

Figure 4-8 Three Primary Networks in Example.com Corporation DIT



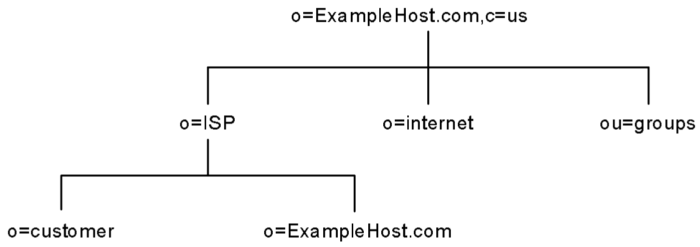
After creating the initial structure of the tree, they create additional branches as illustrated in Figure 4-9:

Figure 4-9 Detailed View of Three Primary Networks in Example.com Corporation DIT



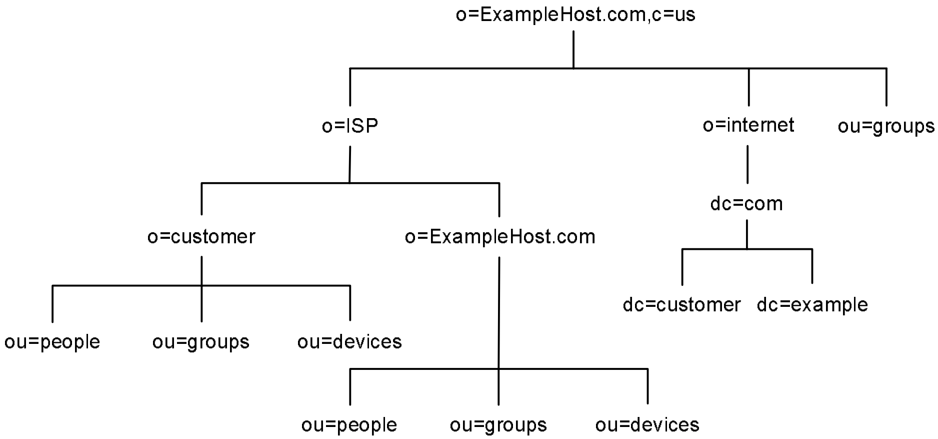
ExampleHost.com, the internet hosting company, branch their directory as illustrated in Figure 4-10:

Figure 4-10 Directory Information Tree for ExampleHost.com



After creating the initial structure of their directory tree, they create additional branches as illustrated in Figure 4-11:

Figure 4-11 Detailed View of the DIT for ExampleHost.com



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 enable certain types of access control. As with replication, it is easier to group similar entries and then administer them from a single branch.

A hierarchical directory tree also enables distributed administration. For example, if you want to 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 also set access controls based on directory content, rather than the directory tree. The ACI filtered target mechanism enables you to 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. To make managing ACIs easier, Directory Server 5.2 enables you to obtain the *effective rights* of an entry, that is, the access control rights that a user has to directory entries and attributes. The effective rights functionality eases user administration, access control policy verification, and debugging. For more information, see “Requesting Effective Rights Information” on page 165.

Distinguished Names, Attributes, and Syntax

This section presents a brief summary of distinguished names, attributes, and syntax information.

Distinguished Names

A *distinguished name* (DN) is the string representation of an entry’s name and location in an LDAP directory. A DN describes a path to a directory entry. Each user and group in your enterprise is represented in Directory Server by a DN. Whenever you make changes to user and group information in the directory, you use distinguished names.

A DN is made up of a number of components called *relative distinguished names* (RDNs). Each RDN identifies a specific entry in the directory. To ensure that every directory entry is unique, LDAP dictates that a single parent entry cannot have two identical RDNs below it.

Usually, a DN for a user or group contains at least three types of RDN:

- A user name, user ID, or group name (identified by the cn or uid keyword)
- An organization name (identified by the o keyword)
- One or more domain name components (identified by the dc keyword). For example, example.com contains two domain name components: example and com.

Other common RDNs are organizational unit (ou), state (st), and country (c).

The composition of a DN depends on the structure of the directory. Most directories are organized by more categories than just country designations and organization names. As a result, the DNs used to identify entries are longer and contain more specific attribute-data pairs. For example, the DNs for three employees or users in the same company might look like this:

```
cn=Ben Hurst, ou=Operations, o=Example Corp, st=CA, c=US
```

```
cn=Jeff Lee, ou=Marketing, o=Example Corp, st=CA, c=US
```

```
cn=Mary Smith, ou=Sales, o=Example Corp, st=MN, c=US
```

In these examples, all three users work in different departments or organizational units (ou) and for the same company or organization (o), Example Corp. The third user works in a different state (st) from the first two users.

LDAP allows organizations and organizational units to contain other organizations and organizational units, enabling the representation of complex enterprises. For example, the DN for a group within a large corporation might look like this:

```
cn=Technical Publications, ou=Super Server Group, ou=Server Division,
o=Example Corporation, o=MegaCorp, dc=megacorp, dc=com
```

Table 4-2 contains a list of common RDN keywords.

Table 4-2 Common RDN Keywords Used in DNs

RDN Keyword	Meaning in a DN	Description

c	country	Country in which the user or group resides. Examples:
		c=US
		c=GB

Table 4-2 Common RDN Keywords Used in DNs *(Continued)*

RDN Keyword	Meaning in a DN	Description

`cn`	common name or full name	Full name of person or object defined by the entry. Examples:
		`cn=Wally Henderson`
		`cn=Database Administrators`
		`cn=printer 3b`

`dc`	domain component	Part of a DNS domain. This keyword is typically used at the top levels of a directory tree. For example, a user in the `example.com` domain might have the following DN: `cn=Barbara Jones,ou=Engineering, dc=example, dc=com`

`l`	locality	Locality in which the user or group resides. This can be the name of a city, country, township, or other geographic regions. Examples:
		`l=Tucson`
		`l=Pacific Northwest`
		`l=Anoka County`

`o`	organization	Organization to which the user or group belongs. Examples:
		`o=Sun Java System Software`
		`o=Public Power & Gas`

`ou`	organizational unit	Unit within an organization. Examples:
		`ou=Sales`
		`ou=Manufacturing`

`sn`	surname	User's last name. Example: `sn=Henderson`

`st`	state or province	State or province in which the user or group resides. Examples:
		`st=Iowa`
		`st=British Columbia`

Attributes

Directory attributes hold descriptive information about an entry. For example, a user entry might have attributes for a user ID, email address, given name, and password.

Table 4-3 contains a list of common user and group directory attributes.

Table 4-3 Common User and Group Directory Attributes

Attribute Keyword	Attribute Name	Description
`givenName`	given name	User's first name.
`mail`	email address	User's or group's email address.
`streetAddress`	street	Street number and address of user or group defined by the entry. Example: `street=12 Main Street`
`telephoneNumber`	telephone	User's or group's telephone number. Example: `(800) 555-9SUN`

`title`	title	User's job title. Examples:
		`title=writer`
		`title=manager`

`uid`	user ID	Name that uniquely identifies the person or object defined by the entry.
`userPassword`	password	A user's password.

A user entry can include many more attributes than those listed above. In addition, you can create new attributes to meet your company's needs. For more information on attributes, see Chapter 3, "Directory Server Schema."

DN and Attribute Guidelines and Syntax

As you create, select, and use directory entries, follow these guidelines:

Escape commas in RDN values (or enclose them in quotation marks.) If an RDN value contains a comma, enclose the part of the name that uses the comma in double-quotation marks or escape it with a backslash. For example, to include the string `Ace Industry, Corp` in a DN, use the form:

```
o="Ace Industry, Corp", c=US
```

You may achieve the same effect using:

```
o=Ace Industry\, Corp, c=US
```

When schema checking is turned on, attributes must match directory schema. If schema checking is turned on, use RDN keywords and attributes that can be recognized by Directory Server and are allowed by the entry’s object classes. If schema checking is turned off, you can use all attributes, regardless of an entry’s object classes. For more information on schema checking, see “Schema Checking” on page 43.

Specify RDNs in the same sequence or path. Remember that a DN represents a path through a directory tree. If RDN keywords are not specified in the appropriate order, Directory Server may not be able to locate an entry. For example,

```
cn=Ralph Swenson, ou=Accounting, o=Example Corp, c=US
```

is not the same as

```
cn=Ralph Swenson, o=Example Corp, ou=Accounting, c=US
```

because the organizational unit (ou) and organization (o) keywords are not listed in the same order.

User IDs must be unique. Exercise caution when using the `ldapmodify` command to create users, since the utility does not check for duplicate user IDs unless an attribute uniqueness plug-in is enabled in the directory for the user ID attribute. For more information, see Chapter 15, “Enforcing Attribute Value Uniqueness,” in the *Directory Server Administration Guide*.

Naming Entries

After designing your directory tree structure, you must 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 attributes you use depend on the type of entry you are naming.

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 against 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 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=example,dc=com
```

While this naming method allows you to instantly recognize the person associated with the entry, the entry might not be unique in an organization where two people have identical names. This leads to a problem known as DN name collisions, which are 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=example,dc=com
```

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

A better method is to identify 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 e-mail address. This option can lead to awkward DNs that include duplicate attribute values (for example: mail=bjensen@Example.com, dc=example,dc=com), so you should use this option only if you cannot find a unique value that you can use with the uid attribute. For example, you could 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 attribute-data pair you use for person entry RDNs, make sure that they are unique, permanent values.

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.

When placing person entries in your directory tree remember:

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

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=Example.com+st=Washington,o=ISP,c=US
```

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

In a hosting environment, 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. For example, if you are naming a group entry, name it as follows:

```
cn=allAdministrators,dc=example,dc=com
```

Sometimes you need to name an entry whose object class does not support the `commonName` attribute. In this case, 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, having identifying attributes visible in the DN simplifies the administration of your directory tree.

Grouping Directory Entries and Managing Attributes

The directory tree organizes the information of entries hierarchically. This hierarchy is a type of grouping mechanism, though it is not well suited for associations between dispersed entries, for organizations that change frequently, or for data that is repeated in many entries. Directory Server provides two additional grouping mechanisms; groups and roles, which offer more flexible associations between entries.

In addition to these grouping mechanisms, Directory Server provides the class of service (CoS) mechanism for managing attributes so that they are shared between entries in a way that is invisible to applications. Like the role mechanism, CoS generates virtual attributes on the entries as they are retrieved. However, CoS does not define membership but rather allows related entries to share data for coherence and space considerations.

These entry grouping and attribute management mechanisms and their associated advantages and limitations are described in the following sections:

- Static and Dynamic Groups
- Managed, Filtered, and Nested Roles
- Role Enumeration and Role Membership Enumeration
- Role Scope

Static and Dynamic Groups

A group is an entry that identifies the other entries that are its members. The scope of possible members of a group is the entire directory, regardless of where the group definition entry is located. Once you know the name of a group, it is easy to retrieve all of its member entries. The following list describes the characteristics of static and dynamic groups, and indicates when it is preferable to use each type:

- Static groups explicitly name their member entries. An entry that defines a static group uses the `groupOfNames` or `groupOfUniqueNames` object class and contains the DN of each member as a value of the `member` or `uniqueMember` attribute respectively. The `member` attribute contains a DN against which the server checks to establish group membership. The `uniqueMember` attribute contains a DN optionally followed by a hash (#) and a unique identifier label, against which the server checks membership.

- Static groups are suitable for groups with few members, such as the group of directory administrators, and not for extremely large groups. You should avoid creating static groups with more than 20,000 members, because they will have very poor performance. For groups of this size and more, you should use dynamic groups or roles. If you must use static groups for more than 20,000 members, use groups of groups rather than a single, large, static group.

- Dynamic groups specify a filter, and all entries that match the filter are members of the group. These groups are dynamic because membership is defined each time the filter is evaluated. The definition entry of a dynamic group belongs to the `groupOfUniqueNames` and `groupOfURLs` object classes. The group members are listed by one or more filters represented as LDAP URL values of the `memberURL` attribute, or by one or more DNs as values of the `uniqueMember` attribute.

NOTE Using the DN of another group as the uniqueMember attribute of a dynamic group enables you to place groups inside other groups. These groups are called *nested* groups.

Although both types of groups can identify members anywhere in the directory, group definitions should be located under an appropriately named node such as ou=Groups. This makes them easy to find, for example, when defining access control instructions (ACIs) that grant or restrict access when the bind credentials are members of a group.

Managed, Filtered, and Nested Roles

Roles are an entry grouping mechanism that enable you to determine role membership as soon as an entry is retrieved from the directory. This overcomes the main disadvantage of the group mechanism. Each role has *members*, or entries that possess the role. Every entry that belongs to a role is given the nsRole virtual attribute whose values are the DNs of all roles for which the entry is a member. As with groups, you can specify role members explicitly or dynamically.

The role mechanism is simple from a client perspective because the directory automatically computes role membership. The nsRole attribute is said to be virtual because it is generated on-the-fly by the server and not stored in the directory. This means that evaluating roles is more resource-intensive than evaluating groups, because the server does the work for the client application. However, checking role membership is uniform and is performed transparently on the server side.

Directory Server supports the following three types of roles:

- Managed Roles - Explicitly assign a role to member entries.
- Filtered Roles - Entries are members if they match a specified LDAP filter. In this way, the role depends upon the attributes contained in each entry.
- Nested Roles - Enable you to create roles that contain other roles.

Managed Roles

Managed roles are similar to static groups, except that membership is defined in each member entry and not in the role definition entry. With managed roles, the administrator assigns a role by adding the `nsRoleDN` attribute to the participating entries. The value of this attribute is the DN of the role definition entry. The static role definition entry only defines the scope of its effect. Members of the role are entries within that scope, that name the DN of the role definition entry in their `nsRoleDN` attribute.

Filtered Roles

Filtered roles are similar to dynamic groups: they define a filter that determines the members of the role. The value of their `nsRoleFilter` attribute defines the filtered role. When the server returns an entry in the scope of a filtered role that matches its filter string, that entry will contain the generated `nsRole` attribute identifying the role.

Nested Roles

Nested roles enable you to create roles that contain other roles. A nested role lists the definition entries of other roles and combines all the members of their roles. If an entry is a member of a role that is listed in a nested role, then the entry is also a member of the nested role.

Role Enumeration and Role Membership Enumeration

Role Enumeration

The `nsRole` attribute is read like any other attribute, and clients may use it to enumerate all roles to which an entry belongs. The `nsRole` attribute can only be used by the roles mechanism and is protected against all modifications. However, it can be read, so if you do not want to expose role membership, you should define access controls to protect it against reading.

Role Membership Enumeration

Because you can perform searches on virtual attributes, so you can search on the `nsRole` attribute and enumerate the members of a role. Note, however, that non-indexed attributes in a search operation may have a considerable performance impact.

Searches based on equality filters are likely to be indexed and as a result efficient, but negation searches will not be indexed and will result in poorer performance. The nsRoleDN attribute is indexed by default so searches on managed roles should be relatively efficient. For filtered and nested roles, where filters can contain both indexed and non-indexed attributes, you should ensure that the filter contains at least one indexed attribute so as not to launch a non-indexed search.

Role Scope

Directory Server provides an attribute that allows the scope of a role to be extended beyond the subtree of the role definition entry. This single-valued attribute, nsRoleScopeDN, contains the DN of the scope to be added to an existing role. The nsRoleScopeDN attribute can only be added to a nested role.

The nsRoleScopeDN attribute enables you to extend the scope of a role in one subtree to include an entry in another subtree. For example, imagine two main subtrees in the Example.com directory tree: o=eng,dc=example,dc=com (the engineering subtree) and o=sales,dc=example,dc=com (the sales subtree.) A user in the engineering subtree requires access to a sales application governed by a role in the sales subtree (SalesAppManagedRole). To extend the scope of the role, you would:

1. Create a role for the user in the engineering subtree, for example, EngineerManagedRole. (This example uses a managed role but it could just as well have been a filtered or nested role).
2. Create a nested role, for example, SalesAppPlusEngNestedRole, in the sales subtree to house the newly created EngineerManagedRole and the initial SalesAppManagedRole.
3. Add the nsRoleScopeDN attribute to the SalesAppPlusEngNestedRole, with the DN of the engineering subtree scope you want to add, (in this case o=eng,dc=example,dc=com.)

The necessary permissions must be granted to the engineering user, so that he can access the SalesAppPlusEngNestedRole role, and in turn the sales application. In addition, the entire scope of the role must be replicated.

NOTE The restriction of extended scope to nested roles means that an administrator who previously managed roles in one domain will only have rights to use the roles that already exist in the other domain, and will not be able to create an arbitrary role in the other domain.

Role Limitations

When creating roles to support your directory service, be aware of the following limitations:

- Roles and chaining

If your directory tree is distributed over several servers using the chaining feature, entries that define roles must be located on the same server as the entries possessing those roles. If one server, A, receives entries from another server, B, through chaining, those entries will contain the roles defined on B, but will not be assigned any of the roles defined on A.

- Filtered Roles cannot use CoS generated attributes

The filter string of a filtered role cannot be based on the values of a CoS virtual attribute. For more information see "About CoS" on page 73. However, the specifier attribute in a CoS definition may reference the `nsRole` attribute generated by a role definition. For information on creating role-based attributes, refer to "Creating Role-Based Attributes" in the *Directory Server Administration Guide*.

- Extending the Scope of Roles

You can extend the scope of roles to different subtrees but they must be on the same server instance. Scoping roles to other servers is not supported.

Deciding Between Groups and Roles

The groups and roles mechanisms provide a degree of overlapping functionality, and both have advantages and disadvantages. Generally, the more recent roles mechanism is designed to provide frequently required functionality more efficiently. Because the choice of a grouping mechanism influences server complexity and determines how clients process membership information, you must plan your grouping mechanism carefully. You should understand the set membership queries and set management operations that will be performed, to decide which mechanism is more suitable.

Advantages of the Groups Mechanism

- Static Groups are preferable to roles for enumerating members, when membership does not exceed 20,000 members.

  If you *only* need to enumerate members of a given set, it is less costly to use static groups, provided that the number of members does not exceed 20,000. Static groups with more than 20,000 members have a negative performance impact. Enumerating members of a static group by retrieving the `member` attribute is easier than recovering all entries that share a role.

- Static groups are preferable to roles for set management operations such as assigning and removing members.

  Static groups are the simplest mechanism for assigning a user to a set or removing a user from a set, because special access rights are not required to add the user to the group.

  Having the right to create the group entry automatically gives you the right to assign members to that group. This is not the case for managed and filtered roles, where the administrator must also have the right to write the `nsroledn` attribute to the user entry. The same access right restrictions also apply indirectly to nested roles, as the ability to create a nested role implies the ability to be able to pull together other roles that have already been defined.

- Dynamic groups are preferable to roles for use in filter-based ACIs.

  If you *only* need to find all members based on a filter, such as for designating bind rules in ACIs, use dynamic groups. Although filtered roles are similar to dynamic groups, they will trigger the roles mechanism and generate the virtual `nsRole` attribute. If your client does not need the `nsRole` value, opting for dynamic groups will avoid the overhead of this computation.

- Groups are preferable to roles for adding or removing sets into or from existing sets.

  If you want to add or remove a set into or from an existing set, the groups mechanism is simplest, as there are no nesting restrictions. The roles mechanism only allows nested roles to receive other roles.

- Groups are preferable to roles if flexibility of scope for grouping entries is critical.

  Groups are flexible in terms of scope as the scope for possible members is the entire directory, regardless of where the group definition entries are located. Although roles can also extend their scope beyond a given subtree, they can only do so by adding a scope-extending attribute `nsRoleScopeDN` to a nested role, which constitutes a scope extension limitation.

Advantages of the Roles Mechanism

- Roles are preferable to groups if you want to enumerate members of aset *and* find all set membership for an entry.

  Roles push membership information out to the user entry where it can be cached to make subsequent membership tests more efficient. The server performs all computations, and the client only needs to read the values of the `nsRole` attribute. In addition, all types of roles appear in this attribute, allowing the client to process all roles uniformly. Roles can perform both operations more efficiently and with simpler clients than is possible with groups.

- Roles are preferable to groups if you want to integrate your grouping mechanism with existing Directory Server functionality such as CoS, Password Policy, Account Inactivation and ACIs.

  If you want to use the membership of a set "naturally" in the server, that is, take advantage of the membership computations that the server will do automatically, roles are a better option. Roles can be used in resource-oriented ACIs, as a basis for CoS, as part of more complex search filters, Password Policy, Account Inactivation, and so forth. Groups do not allow this kind of integration.

Managing Attributes with Class of Service (CoS)

The CoS mechanism enables you to share attributes between entries in a way that is transparent to applications. CoS generates virtual attributes on entries as they are retrieved, in the same way as the roles mechanism. CoS does not define membership, (it does not group entries in the way that the roles mechanism does,) but allows related entries to share data for coherence and space considerations. This section examines the CoS mechanism in more detail and is divided into the following topics:

- About CoS
- Cos Definition Entries and CoS Template Entries
- CoS Priorities
- Pointer CoS, Indirect CoS, and Classic CoS
- CoS Limitations

About CoS

Imagine a directory containing thousands of entries that all have the same value for the `facsimileTelephoneNumber` attribute. Traditionally, to change the fax number, you would update each entry individually, a time consuming job for administrators. Using CoS, the fax number is stored in a single place, and Directory Server automatically generates the `facsimileTelephoneNumber` attribute on every concerned entry as it is returned.

To client applications, a generated CoS attribute is retrieved just as any other attribute. However, directory administrators now have only a single fax value to manage. Also, because there are less values stored in the directory, the database uses less disk space. The CoS mechanism also allows entries to override a generated value or to generate multiple values for the same attribute.

NOTE As CoS virtual attributes are not indexed, referencing them in an LDAP search filter may have an impact on performance.

Generated CoS attributes can be multi-valued. Specifiers may designate several template entries, or there may be several CoS definitions for the same attribute. Alternatively, you can specify template priorities so that only one value is generated from all templates. For more information, refer to “Defining Class of Service (CoS)” in the *Directory Server Administration Guide*. Roles and 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 CoS template. You could use a role-based attribute to set the server look through limit on a role-by-role basis, for example.

CoS functionality can be used recursively; you can generate attributes through CoS that depend on other attributes generated through CoS. Complex CoS schemes may simplify client access to information and ease administration of repeated attributes, but they also increase management complexity and degrade server performance. Avoid overly complex CoS schemes; many indirect CoS schemes can be redefined as classic or pointer CoS, for example.

You should also avoid changing CoS definitions more often than necessary. Modifications to CoS definitions do not take effect immediately, because the server caches CoS information. Although caching accelerates read access to generated attributes, when changes to CoS information occur, the server must reconstruct the cache. This task can take some time, usually in the order of seconds. During cache reconstruction, read operations may still access the old cached information, rather than the newly modified information, which means that if you change CoS definitions too frequently, you are likely to be accessing outdated data.

Cos Definition Entries and CoS Template Entries

The CoS mechanism relies on two types of entries; the CoS definition entry and the CoS template entry.

CoS Definition Entry

The CoS definition entry identifies the type of CoS and the names of the CoS attributes that will be generated. Like the role definition entry, it inherits from the `LDAPsubentry` object class. The location of the definition entry determines the scope of the CoS, which is the entire subtree below the parent of the CoS definition entry. All entries in the branch of the definition entry's parent are called *target entries* for the CoS definition. Multiple definitions may exist for the same CoS attribute, which, as a result, may be multi-valued.

The CoS definition entry is an instance of the `cosSuperDefinition` object class. The CoS definition entry also inherits from one of the following object classes to specify the type of CoS:

- `cosPointerDefinition`
- `cosIndirectDefinition`
- `cosClassicDefinition`

The CoS definition entry contains the attributes specific to each type of CoS for naming the virtual CoS attribute, the template DN, and the specifier attribute in target entries. By default, the CoS mechanism will not override the value of an existing attribute with the same name as the CoS attribute. However, the syntax of the CoS definition entry allows you to control this behavior.

NOTE When schema checking is turned on, the CoS attribute will be generated on all target entries that allow that attribute. When schema checking is turned off, the CoS attribute will be generated on all target entries.

CoS Template Entry

The CoS template entry contains the value that is generated for the CoS attribute. All entries within the scope of the CoS use the values defined here. There may be several templates, each with a different value, in which case the generated attribute may be multi-valued. The CoS mechanism selects one of these values based on the contents of both the definition entry and the target entry.

The CoS template entry is an instance of the `cosTemplate` object class. The CoS template entry contains the value or values of the attributes generated by the CoS mechanism. The template entries for a given CoS are stored in the directory tree at the same level as the CoS definition.

NOTE When possible, definition and template entries should be located in the same place, for easier management. You should also name them in a way that suggests the functionality they provide. For example, a definition entry DN such as `"cn=classicCosGenEmployeeType,ou=People,dc=example,dc=com"` is more descriptive than `"cn=ClassicCos1,ou=People,dc=example,dc=com"`. For more information about the object classes and attributes associated with each type of CoS, refer to " Defining Class of Service (CoS)" in the *Directory Server Administration Guide*.

CoS Priorities

It is possible to create CoS schemes that compete with each other to provide an attribute value. For example, you might have a multi-valued `cosSpecifier` in your CoS definition entry. In such a case, you can specify a template priority on each template entry to determine which template provides the attribute value. Set the template priority using the `cosPriority` attribute. This attribute represents the global priority of a particular template numerically. A priority of zero is the highest possible priority.

Templates that contain no `cosPriority` attribute are considered the lowest possible priority. In the case where two or more templates are considered to supply an attribute value and they have the same (or no) priority, a value is chosen arbitrarily.

Pointer CoS, Indirect CoS, and Classic CoS

There are three types of CoS that differ in how the template, and thus the generated value, is selected. The three different types of CoS are:

- Pointer CoS
- Indirect CoS
- Classic CoS

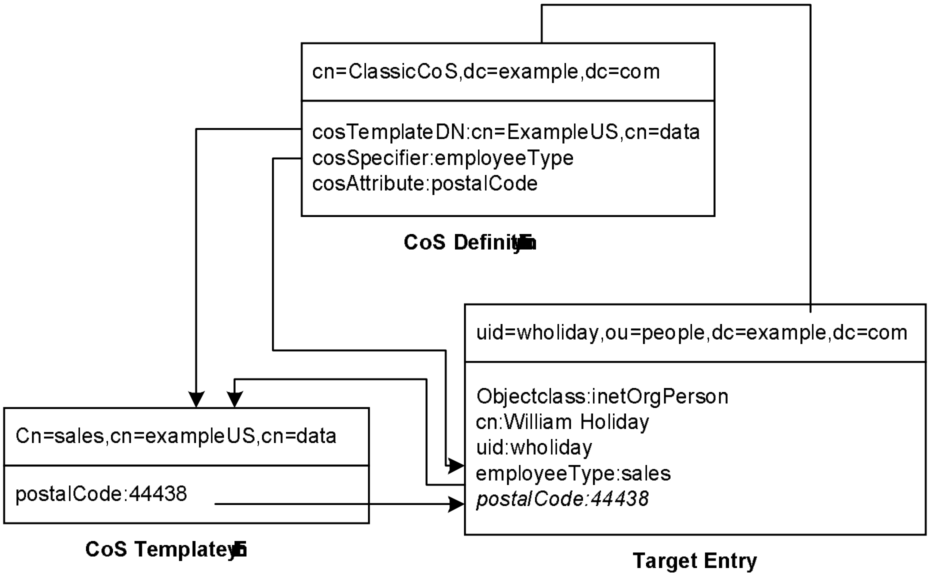
Pointer CoS

Pointer CoS is the simplest type of CoS. The pointer CoS definition entry provides the DN of a specific template entry of the `cosTemplate` object class. All target entries have the same CoS attribute value, as defined by this template.

Pointer CoS Example

Figure 4-12 shows a pointer CoS that defines a common postal code for all of the entries stored under `dc=example,dc=com`. The CoS definition entry, CoS template entry and target entry are indicated:

Figure 4-12 Example of a Pointer CoS Definition and Template


cn=ClassicCoS,dc=example,dc=com
cosTemplateDN:cn=ExampleUS,cn=data
cosSpecifier:employeeType
cosAttribute:postalCode
CoS Definityn
uid=wholiday,ou=people,dc=example,dc=com
Objectclass:inetOrgPerson
cn:William Holiday
uid:wholiday
employeeType:sales
postalCode:44438
Cn=sales,cn=exampleUS,cn=data
postalCode:44438
CoS Templatey
Target Entry

The template entry is identified by its DN, `cn=exampleUS,cn=data`, in the CoS definition entry. Each time the `postalCode` attribute is queried on entries under `dc=example,dc=com`, Directory Server returns the value available in the template entry `cn=exampleUS,cn=data`. Therefore, the postal code will appear with the entry `uid=wholiday,ou=people,dc=example,dc=com`, but it is not stored there.

In a scenario where several shared attributes are generated by CoS for thousands or millions of entries, instead of existing as real attributes in each entry, the storage space savings and performance gains provided by CoS are considerable.

Indirect CoS

Indirect CoS allows any entry in the directory to be a template and provide the CoS value. The indirect CoS definition entry identifies an attribute, called the indirect specifier, whose value in a target entry determines the template used for that entry. The indirect specifier attribute in the target entry must contain a DN. With indirect CoS, each target entry may use a different template and thus have a different value for the CoS attribute.

For example, an indirect CoS that generates the `departmentNumber` attribute may use an employee's manager as the specifier. When retrieving a target entry, the CoS mechanism will use the DN value of the `manager` attribute as the template. It will then generate the `departmentNumber` attribute for the employee using the same value as the manager's department number.

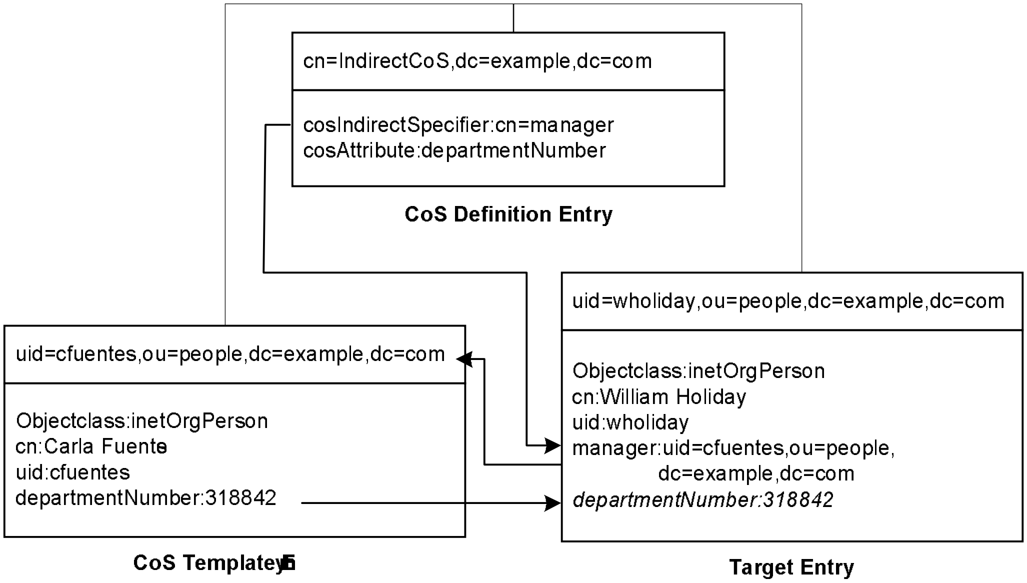
CAUTION Because templates may be arbitrary entries anywhere in the directory tree, implementing access control for indirect CoS can become extremely complex. In deployments where performance is critical, you should also avoid overusing indirect CoS due to its resource intensive nature.

In many cases, results that are similar to those made possible by indirect CoS can be achieved by limiting the location of the target entries with classic CoS or using the less flexible pointer CoS mechanism.

Indirect CoS Example

Figure 4-13 on page 78 shows an indirect CoS that uses the `manager` attribute of the target entry to identify the template entry. In this way, the CoS mechanism can generate the `departmentNumber` attribute of all employees to be the same as their manager's, ensuring that it is always up to date.

Figure 4-13 Example of an Indirect CoS Definition and Template


cn=IndirectCoS,dc=example,dc=com
cosIndirectSpecifier:cn=manager
cosAttribute:departmentNumber
CoS Definition Entry
uid=wholiday,ou=people,dc=example,dc=com
Objectclass:inetOrgPerson
cn:William Holiday
uid:wholiday
manager:uid=cfuentes,ou=people,
dc=example,dc=com
departmentNumber:318842
Target Entry
uid=cfuentes,ou=people,dc=example,dc=com
Objectclass:inetOrgPerson
cn:Carla Fuente
uid:cfuentes
departmentNumber:318842
CoS Templatey5

The indirect CoS definition entry names the specifier attribute, which in this example, is the `manager` attribute. William Holiday's entry is one of the target entries of this CoS, and his `manager` attribute contains the DN of `uid=cfuentes,ou=people,dc=example,dc=com`. Therefore, Carla Fuentes' entry is the template, which in turn provides the `departmentNumber` attribute value of `318842`.

Classic CoS

Classic CoS combines the pointer and indirect CoS behavior. The classic CoS definition entry identifies the base DN of the template and a specifier attribute. The value of the specifier attribute in the target entries is then used to construct the DN of the template entry as follows:

`cn=`*specifierValue*, *baseDN*

The template containing the CoS values is determined by the combination of the RDN (relative distinguished name) value of the specifier attribute in the target entry and the template's base DN.

Classic CoS templates are entries of the `cosTemplate` object class to avoid the performance issue associated with arbitrary indirect CoS templates.

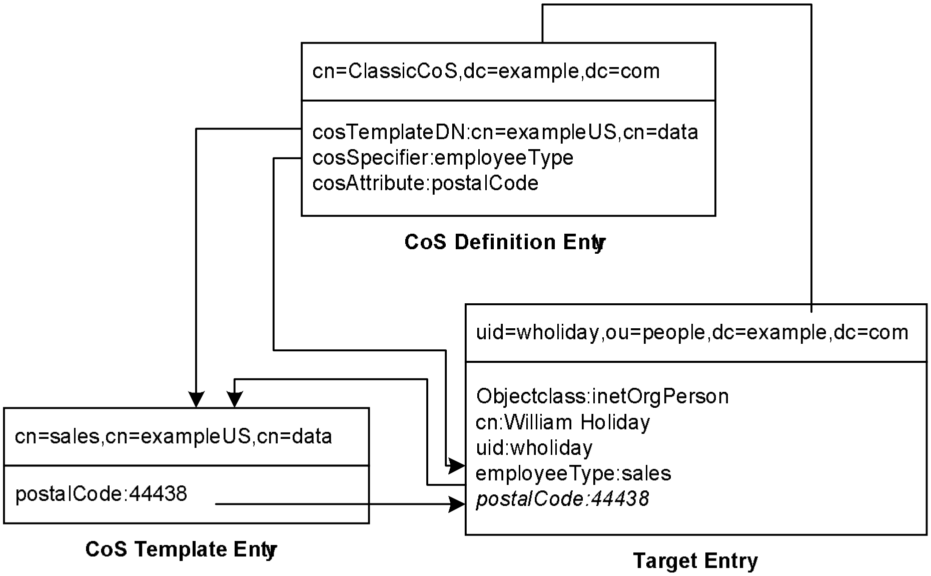
Classic CoS Example

The classic CoS mechanism determines the DN of the template from the base DN given in the definition entry and the specifier attribute in the target entry. The value of the specifier attribute is taken as the `cn` value in the template DN. Template DNs for classic CoS must therefore have the following structure:

`cn=`*specifierValue*`,` *baseDN*

Figure 4-14 shows a classis CoS definition that generates a value for the postal code attribute:

Figure 4-14 Example of a Classic CoS Definition and Template


cn=ClassicCoS,dc=example,dc=com
cosTemplateDN:cn=exampleUS,cn=data
cosSpecifier:employeeType
cosAttribute:postalCode
CoS Definition Enty
uid=wholiday,ou=people,dc=example,dc=com
Objectclass:inetOrgPerson
cn:William Holiday
uid:wholiday
employeeType:sales
postalCode:44438
cn=sales,cn=exampleUS,cn=data
postalCode:44438
CoS Template Enty
Target Entry

In this example, the Cos definition entry's `cosSpecifier` attribute names the `employeeType` attribute. The combination of this attribute and the template DN identifies the template entry as `cn=sales,cn=exampleUS,cn=data`. The template entry then provides the value of the `postalCode` attribute to the target entry.

CoS Limitations

The CoS functionality is a complex mechanism which, for performance and security reasons, is subject to the following limitations.

- Restricted subtrees

  You cannot create CoS definitions in either the `cn=config` or `cn=schema` subtrees.

- Unindexed searches

  Searches in suffixes where an attribute is declared as a CoS-generated attribute will result in an unindexed search. This may have a significant impact on performance. In suffixes where the same attribute is NOT declared as a CoS attribute, the search will be indexed.

- Restricted attribute types

  The following attributes should not be generated by CoS because they do not have the same behavior as real attributes of the same name:

  - `userPassword` - A CoS-generated password value cannot be used to bind to Directory Server.
  - `aci` - Directory Server will not apply any access control based on the contents of a virtual ACI value defined by CoS.
  - `objectclass` - Directory Server will not perform schema checking on the value of a virtual object class defined by CoS.
  - `nsRoleDN` - A CoS-generated `nsRoleDN` value will not be used by the server to generate roles.

- All templates must be local

  The DNs of template entries, either in a CoS definition or in the specifier of the target entry, must refer to local entries in the directory. Templates and the values they contain cannot be retrieved through directory chaining or referrals.

- CoS virtual values cannot be combined with real values

  The values of a CoS attribute are never a combination of real values from the entry and virtual values from the templates. When the CoS overrides a real attribute value, it replaces all real values with those from the templates. However, the CoS mechanism can combine virtual values from several CoS definition entries. For more information, see “CoS Limitations” in the *Directory Server Administration Guide*.

- Filtered roles cannot use CoS-generated attributes

The filter string of a filtered role cannot be based on the values of a CoS virtual attribute. However, the specifier attribute in a CoS definition may reference the `nsRole` attribute generated by a role definition. For more information, see "Creating Role-Based Attributes" in the *Directory Server Administration Guide*.

- Access Control Instructions (ACIs)

  The server controls access to attributes generated by a CoS in exactly the same way as regular, stored attributes. However, access control rules that depend on the value of attributes generated by CoS are subject to the conditions described in "Restricted attribute types" on page 80.

- CoS cache latency

  The CoS cache is an internal Directory Server structure that keeps all CoS data in memory to improve performance. This cache is optimized for retrieving CoS data to be used in computing virtual attributes, even while CoS definition and template entries are being updated. Therefore, once definition and template entries have been added or modified, there may be a slight delay before they are taken into account. This delay depends on the number and complexity of CoS definitions, as well as the current server load, but it is usually in the order of a few seconds. Consider this latency before designing overly complex CoS configurations.

Other Directory Tree Resources

The following links provide additional information on 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

Distribution, Chaining, and Referrals

Chapter 4, “The Directory Information Tree,” described how Directory Server stores entries. Because Directory Server can store a large number of entries, you may need to distribute entries across more than one server. The directory topology describes how you divide your directory tree among multiple physical Directory Servers, and how these servers link with one another.

This chapter describes how you can use data distribution, chaining, and referrals to manage directory data more effectively. It is divided into the following topics:

- Topology Overview
- Distributing Data
- Referrals and Chaining

Topology Overview

A distributed directory is one in which the directory tree is spread across multiple physical Directory Servers. Dividing your directory in this way enables you to:

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

When a directory is divided among several servers, each server is responsible for only a part of the directory tree. The distributed directory works in a similar way to the Domain Name Service (DNS), which assigns each portion of the DNS namespace to a particular DNS server. In the same way, you can distribute your directory namespace across servers while maintaining, from a client point of view, a single directory tree.

Directory Server also provides the *referral* and *chaining* mechanisms for linking directory data stored in different databases. (The suffix is the basic unit for tasks such as replication, performing backups, and restoring data.) The remainder of this chapter describes suffixes, referrals, and chaining, and describes how you can design indexes to improve directory performance.

Distributing Data

Distributing data enables you to scale your directory across multiple server instances, without the need for all directory entries to be located on each server in your enterprise. The server instances may or may not (depending on performance requirements) be stored on several machines. A distributed directory can therefore 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 distribution details from the user or client application. As far as directory clients are concerned, a single directory answers their directory queries.

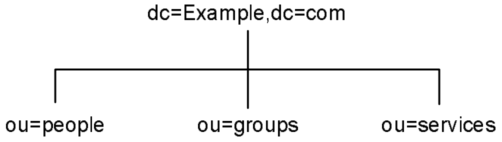
The following sections describe the mechanics of data distribution in more detail:

- Using Multiple Databases
- About Suffixes

Using Multiple Databases

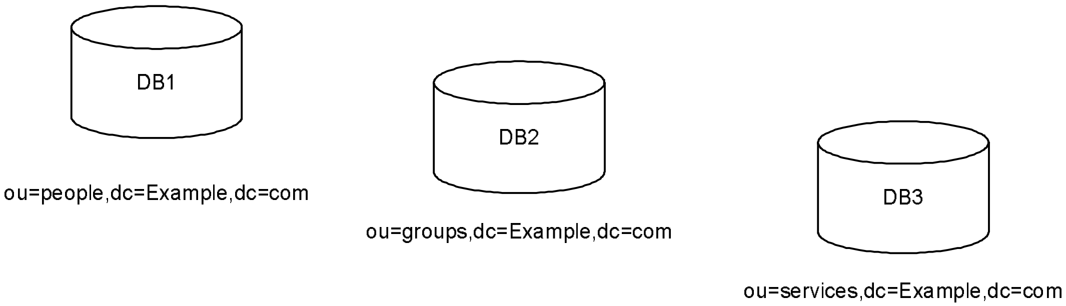
Directory Server stores data in LDBM databases. The LDBM database is a high-performance disk-based database. Each database consists of a set of large files that contains all of the data assigned to it. You can store different portions of your directory tree in different databases. Imagine, for example, your directory tree contains three subsuffixes, as shown in Figure 5-1:

Figure 5-1 Directory Tree With Three Subsuffixes



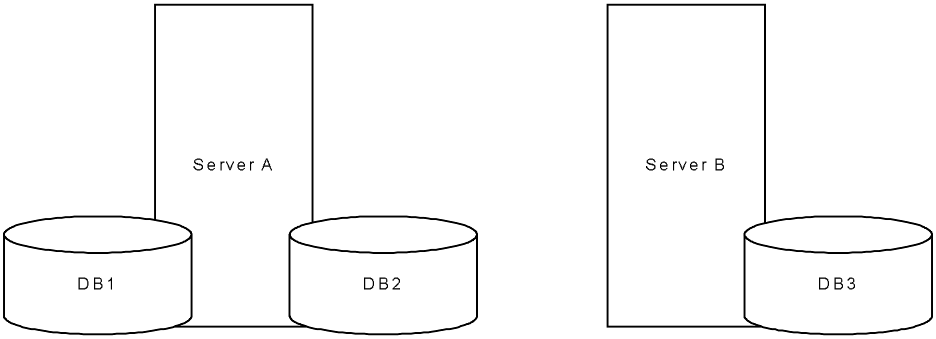
You can store the data of the three subsuffixes in three separate databases as shown in Figure 5-2:

Figure 5-2 Three Subsuffixes Stored in Three Separate Databases


DB1
ou=people,dc=Example,dc=com
DB2
ou=groups,dc=Example,dc=com
DB3
ou=services,dc=Example,dc=com

When you divide your directory tree among a number of databases, these databases can be distributed across multiple servers, which generally equates to several physical machines to improve performance. The three databases in the preceeding figure can be stored on two servers as shown in Figure 5-3:

Figure 5-3 Example.com's Three Databases Stored on Two Servers A and B



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. Directory Server also supports adding databases dynamically, so you can add new databases as required, without taking the entire directory off-line.

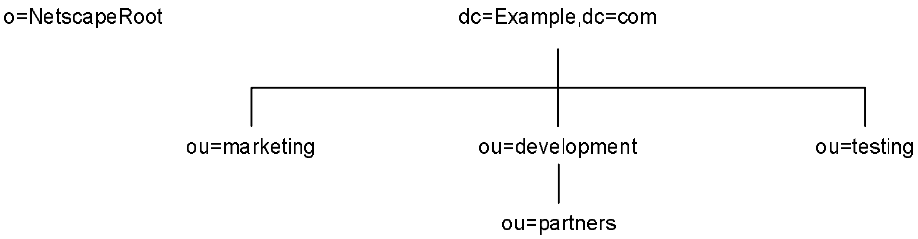
About Suffixes

Each database contains the data within a *suffix* of Directory Server. You can create both suffixes and subsuffixes to organize the contents of your directory tree. A suffix is the entry at the root of a tree. It can be the root of the entire directory tree, or part of a larger tree.

A subsuffix is a branch underneath a suffix. Subsuffixes represent the distribution of directory data.

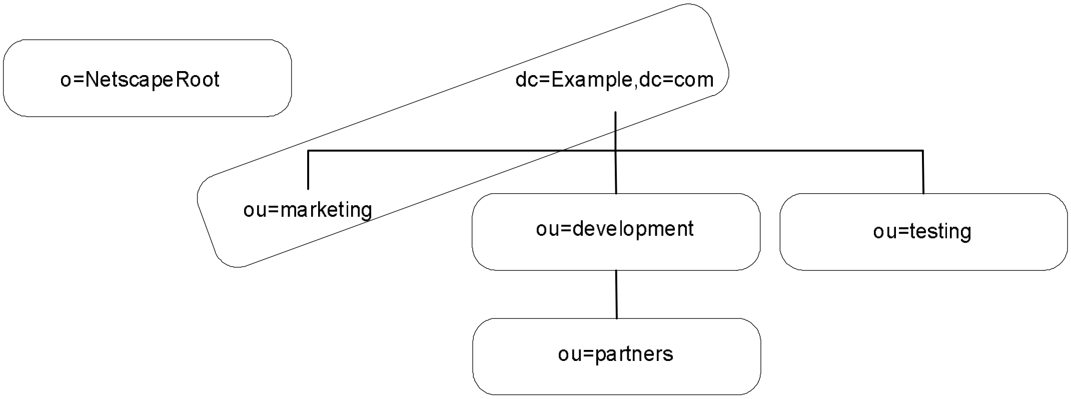
For example, the directory tree for Example.com is shown in Figure 5-4:

Figure 5-4 Example.com Directory Tree



Example.com decides to split their directory tree across five different databases, as illustrated in Figure 5-5:

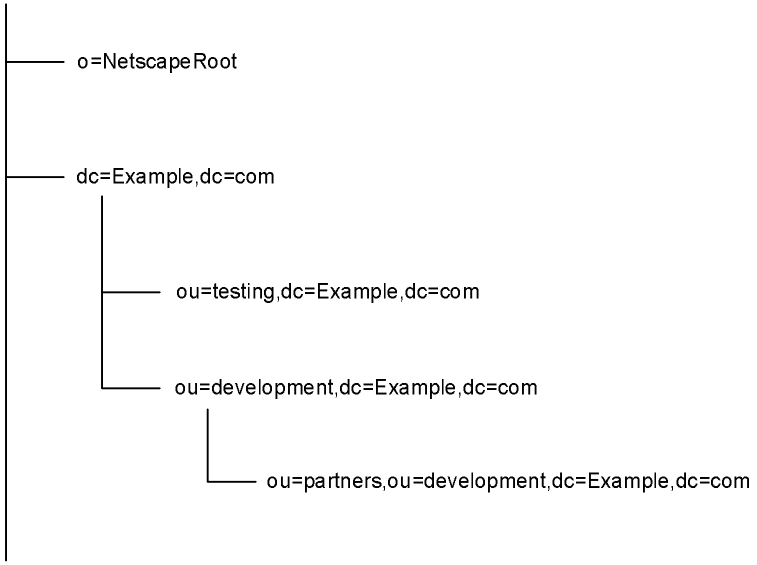
Figure 5-5 Example.com Corporation's Directory Tree Split Across Five Databases



o=NetscapeRoot and dc=Example,dc=com are both suffixes. ou=testing,dc=Example,dc=com, ou=development,dc=Example,dc=com, and ou=partners,ou=development,dc=Example,dc=com are subsuffixes of the dc=Example,dc=com suffix. The suffix dc=Example,dc=com contains the data in the ou=marketing branch of the original directory tree.

The suffixes and subsuffixes that result from this division contain entries as shown in Figure 5-6:

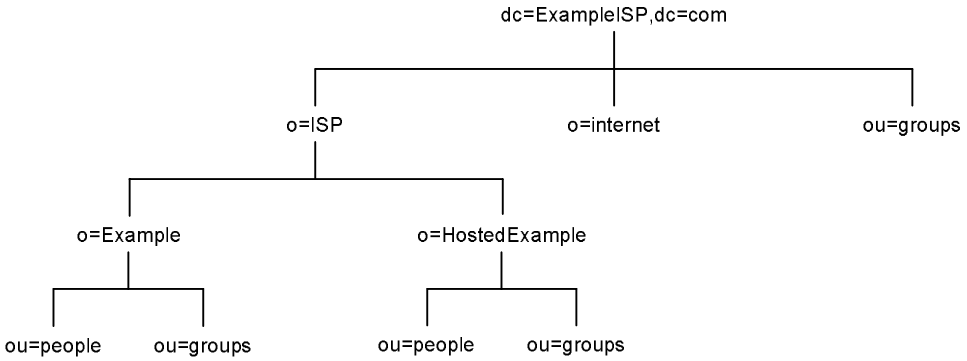
Figure 5-6 Example.com Corporation Suffixes and Associated Entries


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

Your directory might contain more than one suffix. For example, an ISP, ExampleISP.com might host several websites, one for its own website ExampleISP.com and one for another website called HostedExample.com. The ISP can choose between creating one suffix, which houses everything, or two suffixes to separate the hosted part of the organization from internal ExampleISP.com data.

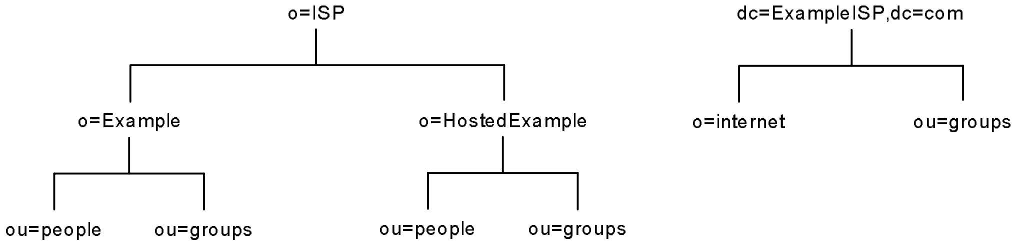
The first solution with just one suffix for all data, would result in a directory information tree as shown in Figure 5-7 on page 88:

Figure 5-7 ExampleISP.com Directory Tree with One Suffix



If the ISP created two suffixes, one corresponding to its own naming context, and one corresponding to the organizations it hosts, the directory information tree would appear as follows:

Figure 5-8 ExampleISP.com's Directory Tree With Two Suffixes



The entries for each hosted organization (`o=Example` and `o=HostedExample`) are subsuffixes of the `o=ISP` suffix, with the `ou=people` and the `ou=groups` branches as subsuffixes of each hosted organization.

Referrals and Chaining

When data is distributed over several suffixes, you must define the relationships between the distributed data. You do this using pointers to directory information held in different suffixes. Directory Server provides the referral and chaining mechanisms to link 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 completing the operation.

The following sections describe and compare these two mechanisms.

Using Referrals

A referral is a piece of information returned by a server that tells a client application which server to contact to proceed with an operation request. Directory Server supports three types of referrals:

- 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 configured at the server level using the `nsslapd-referral` attribute.

- Suffix Referrals

  When an entire suffix has been taken offline for maintenance or security reasons, the server will return the referrals defined by that suffix. Read-only replicas of a suffix also return referrals to the master server when a client requests a write operation.

- 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 the three referral types supported by Directory Server.

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

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

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

The LDAP client application 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 Directory Server (such as the command-line utilities) automatically follow the referral. The same bind credentials you supply on the initial server request are used to access the referred 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.

Default Referrals

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, Directory Server returns a default referral.

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

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

However, the server manages only entries stored under the `dc=europe,dc=Example,dc=com` suffix. The directory returns a referral to the client that indicates which server to contact for entries stored in the `dc=Example,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, stored in the `dse.ldif` configuration file.

For information on configuring default referrals, see “Setting the Default Referrals” in Chapter 2 of the *Directory Server Administration Guide*.

Suffix Referrals

If you want to limit access to a suffix without disabling it completely, you can modify the access permissions to allow read-only access. In this case you must define a suffix referral to another server for write operations. You can also deny both read and write access and define a referral for all operations on the suffix. Suffix referrals can also be used to temporarily point a client application to a different server. For example, you might add a referral to a suffix so that the suffix points to a different server while backing up the contents of the suffix.

Imagine you have two major sites in the US, one based in New York and the other in Los Angeles. A client application sends a query which concerns the New York site as follows:

```
uid=bjensen,ou=people,dc=US,dc=Example,dc=com
```

You can configure a suffix referral to `dc=NewYork,dc=US,dc=Example,dc=com` so that the request is processed by the suffix that contains the `dc=NewYork` subtree.

Suffix referrals are configured with the `nsslapd-state` and `nsslapd-referral` attributes in the mapping tree entry for that suffix. The `nsslapd-referral` attribute specifies the LDAP URL(s) to be returned by the suffix. The `nsslapd-state` attribute can take one of four values:

- `nsslapd-state: backend` where the suffix processes all operations.
- `nsslapd-state: disabled` where the suffix is not available for processing and an error is returned in response to requests made by client applications.
- `nsslapd-state: referral` where a referral is returned for all requests made to this suffix.

- `nsslapd-state: referral on update`where the suffix is used for all operations except update requests, which receive a referral. The `referral on update` state is used internally by the server when replication is configured, to prevent consumers from processing update requests. However, you can also use this state to restrict access to read operations on certain suffixes for load balancing or performance.

For information on configuring suffix referrals, see Setting Access Permissions and Referrals in Chapter 3 of the *Directory Server Administration Guide*.

Smart Referrals

Directory Server also supports *smart referrals*, which enable you to associate a directory entry or directory tree to a specific LDAP URL. Associating directory entries to specific LDAP URLs enables 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, smart referrals are stored within the directory itself.

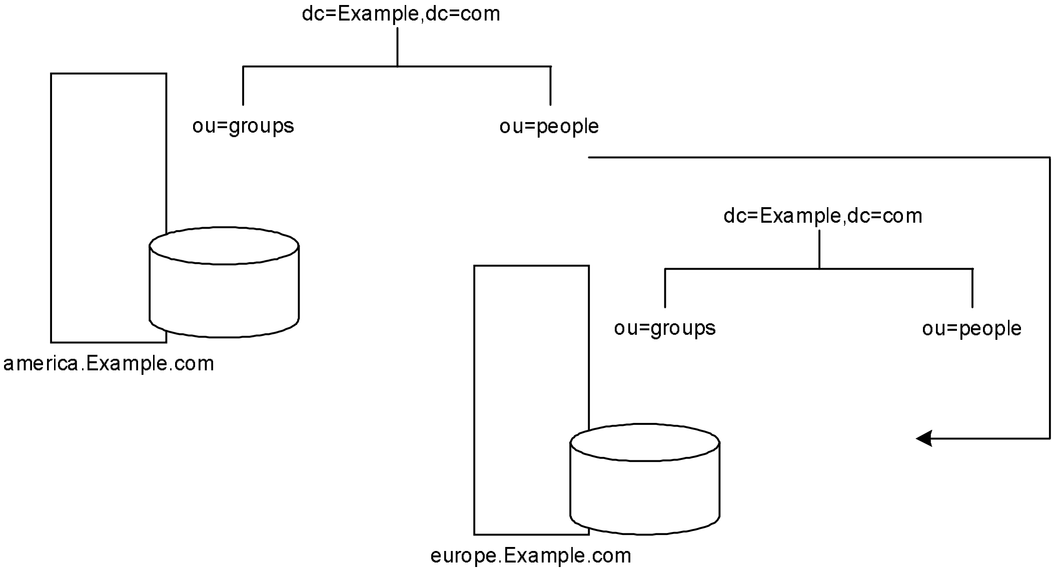
For example, the directory for the American office of Example.com contains the following directory branch point: `ou=people,dc=Example,dc=com`.

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

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

Any requests made to the people branch of the American directory are redirected to the European directory. An illustration of this smart referral is shown in Figure 5-9 on page 93:

Figure 5-9 Smart Referral From American Directory to European Directory



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 Example.com makes a request to the European directory for the phone number of an Example.com employee in America. The referral returned by the directory is:

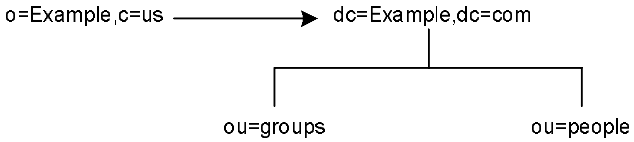
```
ldap://europe.Example.com:389/ou=US employees,dc=Example,dc=com
```

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=Example,c=us` to `dc=Example,dc=com`, you would put the following smart referral on the `o=Example,c=us` entry:

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

as illustrated in Figure 5-10:

Figure 5-10 Smart Referral Traffic



Because you are redirecting queries from one namespace to another on the same machine, there is no need to provide the `host:port` information pair which usually appears in the URL. Because this pair is empty in the URL, the URL pointing to the same Directory Server contains three slashes.

NOTE To make best use of referrals, do not make the base of your search below where the referral is configured.

For more information on LDAP URLs see “LDAP URL Reference” in the *Directory Server Administration Reference*. For more information on how to include smart URLs on Directory Server entries, see “Creating Smart Referrals” in Chapter 2 of the *Directory Server Administration Guide*.

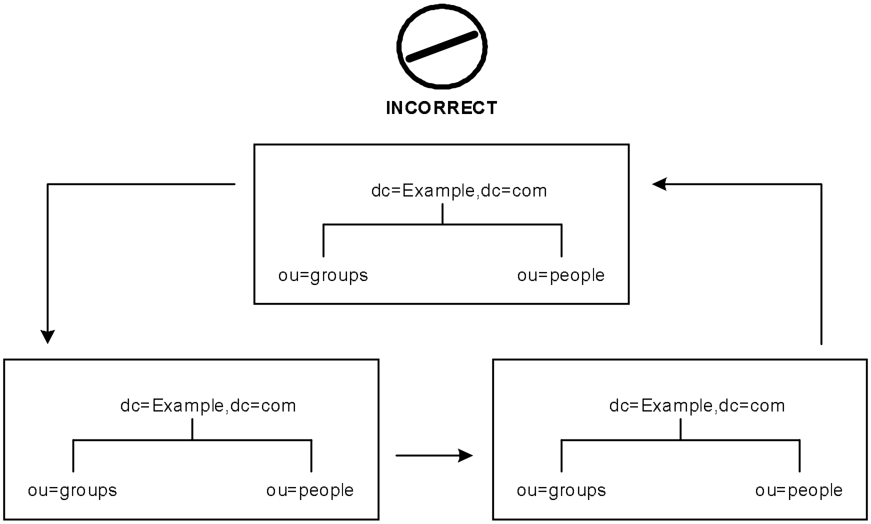
Tips for Designing Smart Referrals

Consider the following points when using smart referrals:

- Keep the design simple.

  A complex web of referrals makes directory administration difficult. Also, overusing smart referrals can lead to circular referral patterns, in which a referral points to an LDAP URL, which in turn points to another LDAP URL, and so on until a referral somewhere in the chain points back to the original server. A circular referral pattern is depicted in Figure 5-11 on page 95:

Figure 5-11 Incorrect Circular Referral Pattern Caused by the Overuse of Smart Referrals



- Redirect at major branch points.

  Limit referrals to handling redirection at the suffix level. 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 directory structure that is difficult to secure. 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, and reduce 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 be refused access.

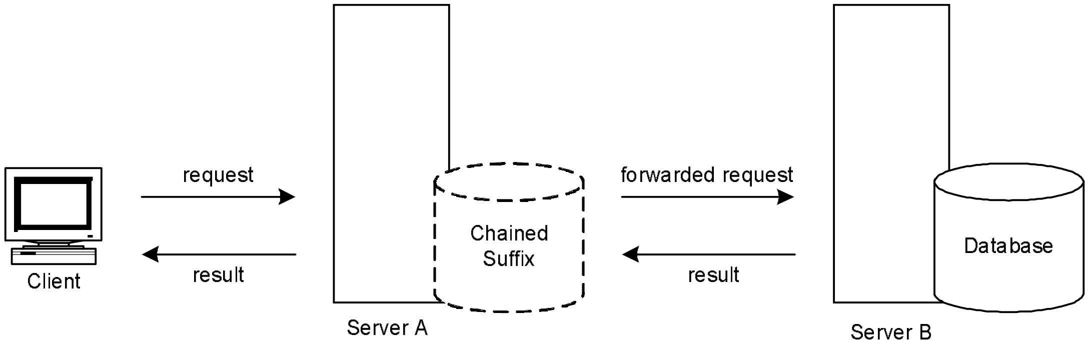
  Also, the client credentials must be available on the server to which the client is referred for client authentication to take place.

Using Chaining

Chaining is a method for relaying requests to another server. This method is implemented through chained suffixes. As described in "Distributing Data" on page 84, a chained suffix 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 chained suffix, 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 Figure 5-12.

Figure 5-12 Chaining Operation



Each chained suffix is associated to a remote server holding data. You can also configure alternate remote servers containing replicas of the data for the chained suffix to use when there is a failure. For more information on configuring chained suffixes, refer to "Creating Chained Suffixes" in Chapter 3 of the *Directory Server Administration Guide*.

Chained suffixes provide the following features:

- Invisible access to remote data.

  Because the chained suffix 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 chained suffix 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 chained suffix 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 regarding access control and chained suffixes see “Access Control Through Chained Suffixes” in Chapter 3 of the *Directory Server Administration Guide*.

Deciding Between Referrals and Chaining

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

The main difference between using referrals and using chaining 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.

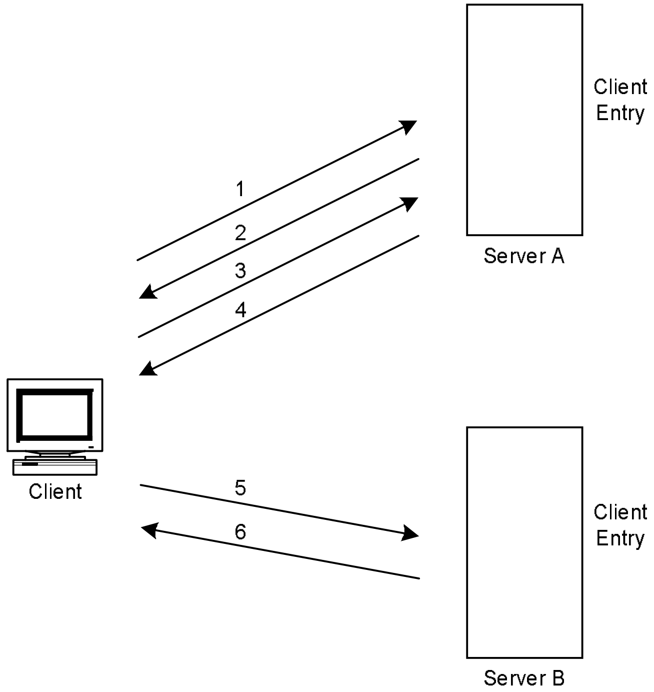
Also, with referrals a client must authenticate, meaning that the servers to which clients are being referred need to contain the client credentials. With chaining, client authentication takes place only once. Clients do not need to authenticate again on the servers to which their requests are chained.

Evaluating Access Controls

Chaining evaluates access controls differently from referrals. With referrals, a bind DN entry 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. Figure 5-13 on page 99 shows how the operation would work using referrals:

Figure 5-13 Client Application Search Request Redirected Through a Referral



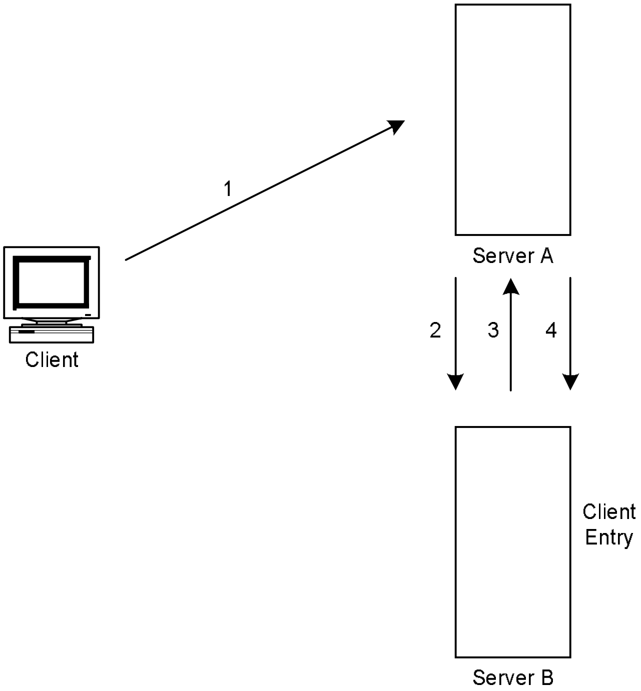
In this figure, the client application performs the following steps:

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 using chaining would work as shown in Figure 5-14:

Figure 5-14 Search Request using Chaining



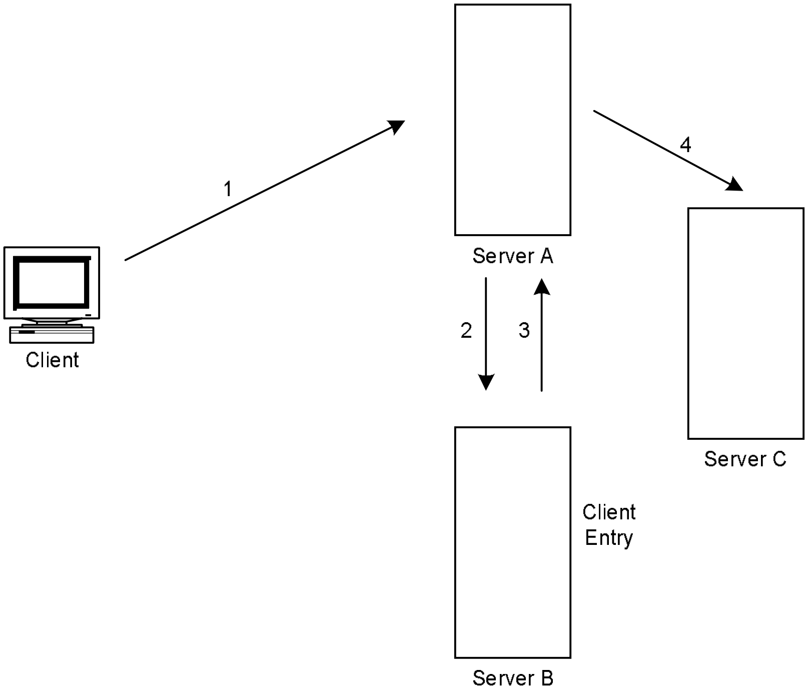
In this figure, the following steps are performed:

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 chained suffix 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 chained suffix. The chained suffix 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. Figure 5-15 illustrates how two chained suffixes can be used to satisfy a client's search request:

Figure 5-15 Chaining Using Two Chained Suffixes to Process a Client's Search Request



In Figure 5-15, 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 chained suffix 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 chained suffix. The chained suffix contacts a remote data store located on Server C to process the search operation.

However, chained suffixes 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 chained suffix impersonates the client when it contacts remote servers. If the remote database contains IP-based access controls, it will evaluate them using the chained suffix's domain rather than the original client domain.

Chapter 6

Understanding Replication

Replicating directory contents increases the availability of your directory. It can also assist in increasing global search performance if additional measures such as load balancing are implemented. Although replication can increase write availability, it does not increase write or update performance.

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

This chapter discusses the use of replication in your deployment, and contains the following topics:

- Introduction to Replication
- Common Replication Configurations
- Defining a Replication Strategy
- Using Replication with Other Directory Features
- Replication Monitoring

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 suffix) between servers, except the configuration or monitoring information subtrees.

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

- Fault tolerance and failover

  By replicating directory trees to multiple servers, you can ensure that your directory is available even if a hardware, software, or network problem prevents directory client applications from accessing a particular Directory Server. Clients can be referred to another directory for read and write operations. Note that to support write failover, you must have more than one master copy of your data in the replication environment.

- Reduced response time by load balancing

  By replicating your directory tree across servers, you can reduce the access load on a given machine, thereby improving server response time. Replication is *not* a solution for write scalability. To increase the write scalability in your deployment, you must consider data partitioning.

- Reduced response time by localizing data

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

- Local data management

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

Before defining a replication strategy, you should have a basic understanding of how replication works. This section includes an overview of:

- Replication Concepts
- Data Consistency

Replication Concepts

When considering implementing replication, start by answering the following fundamental questions:

- What information do you want to replicate?
- Which server or servers hold the master copy of that information?
- Which server or servers hold read-only copies of the information?
- What should happen when a read-only server receives modification requests from client applications; that is, to which server should requests be referred?

These decisions cannot be made effectively without an understanding of how Directory Server implements replication. For example, when you decide what information you want to replicate, you need to know the smallest replication unit that Directory Server can handle. The following sections explain the replication concepts as implemented in Directory Server.

Replica

A database that participates in replication is defined as a *replica*. There are three kinds of replicas:

- Master replica or read-write replica: a read-write database that contains a master copy of the directory data. A master replica can process update requests from directory clients.
- Consumer replica: a read-only database that contains a copy of the information held in the master replica. A consumer replica can process search requests from directory clients but refers update requests to master replicas.
- Hub replica: a read-only database, like a consumer replica, but stored on a Directory Server that *supplies* one or more consumer replicas.

You can configure Directory Server to manage several replicas. Each replica can have a different role in replication.

Unit of Replication

The smallest unit of replication is the suffix. The replication mechanism requires that one suffix correspond to one database. This means that you cannot replicate a suffix (or namespace) that is distributed over two or more databases using custom distribution logic. The unit of replication applies to both consumers and suppliers, which means that you cannot replicate two suffixes to a consumer holding only one suffix, and vice versa.

Replica ID

Master replicas require a unique replica identifier (ID) while consumer replicas all have the same replica ID. The replica ID for masters can be any 16 bit integer between 1 and 65534. Consumer replicas all have the replica ID of 65535. The replica ID identifies the replica on which changes were made, enabling the changes to be replicated correctly.

If a server hosts several replicas (or suffixes,) the replicas may all have the same replica ID, provided that the replica ID is unique between the masters of a single, replicated suffix. Using the same replica ID for all the suffixes on a master enables you to associate a master with only one replica ID independently of the suffixes.

Suppliers and Consumers

A Directory Server that replicates to other servers is called a *supplier*. A Directory Server that is updated by other servers is called a *consumer*. The supplier replays all updates on the consumer through specially designed LDAP v3 extended operations. In terms of performance, a supplier is therefore likely to be a demanding client application for the consumer.

In some cases a server can be both a supplier and a consumer. This is true in the following cases:

- When the server contains a hub replica; that is, it receives updates from a supplier and replicates the changes to consumer(s). For more information, refer to "Cascading Replication" on page 120.
- In multi-master replication, when a master replica is mastered on two different Directory Servers, each server acts as a supplier and a consumer of the other server. For more information, refer to "Multi-Master Replication" on page 115.
- When the server manages a combination of master replicas and consumer replicas.

A server that plays the role of a consumer only (that is, it contains only a consumer replica) is called a *dedicated consumer*.

In Directory Server, replication is always initiated by the supplier, never by the consumer. This is called supplier-initiated replication, as suppliers push the data to consumers. Earlier versions of Directory Server allowed consumer-initiated replication, in which consumers could be configured to pull data from suppliers. From Directory Server 5.0, this has been replaced by a procedure in which the consumer can prompt the supplier to send updates.

For a master replica, the server must:

- Respond to update (add, delete, modify, or modrdn) requests from directory clients.
- Maintain historical information and a change log for the replica.
- Initiate replication to consumers.

  The server containing the master replica is responsible for recording the changes made to the master replicas it manages. It makes sure that any changes are replicated to consumers.

For a hub replica, the server must:

- Respond to read requests.
- Refer update requests to the servers that contain a master replica.
- Maintain historical information and a change log for the replica.
- Initiate replication to consumers.

For a consumer replica, the server must:

- Respond to read requests.
- Maintain historical information for the replica.
- Refer update requests to the servers that contain a master replica.

  Whenever a request to add, delete, or change an entry is received by a consumer, the request is referred via the client to the server, or servers, that contain the master replica; that is, the server acting as the supplier in the replication flow. The supplier performs the request, then replicates the change.

  It is possible to configure consumer or hub replicas not to return a referral, but to return an error instead (if this is required for security or performance reasons.) See "Referrals" on page 108 for more information.

Online Replica Promotion and Demotion

Directory Server 5.2 enables you to promote and demote replicas online. Promoting or demoting a replica changes its role in the replication topology. Dedicated consumers may be promoted to hubs, and hubs may be promoted to masters. Masters may be demoted to hubs, and hubs may be demoted to dedicated consumers. To promote a consumer replica to a master replica, you need to promote it first to a hub replica and then to a master replica. The same incremental approach applies to online demotion. For more information see "Promoting and Demoting Replicas," in Chapter 8 of the *Directory Server Administration Guide*.

In addition to providing increased flexibility, online replica promotion and demotion provides increased failover capabilities. Imagine, for example, a two-way multi-master replication scenario, with two hubs configured for additional load balancing and failover. Should one of the masters go offline, you would simply need to promote one of the hubs to a master to maintain optimal read-write availability. When the master replica came back online, a simple demotion back to a hub replica would return you to the original topology.

NOTE Once a hub is demoted to a consumer, the replica is no longer able to propagate changes (as a consumer it will not have a change log.) Before demoting a hub to a consumer, you must therefore verify that the hub is synchronized with the other servers. To ensure this, you can use the replication monitoring tool `insync` (see "Replication Monitoring" on page 140 for more information.)

Referrals

When a consumer receives a modification request, it *does not* forward the modification request to the server that contains the master replica. Instead, it returns a list containing the URLs of the possible masters that could satisfy the client's modification request. These URLs are referrals.

The replication mechanism automatically configures consumers to return referrals for all known masters in the replication topology. However, you can also add your own referrals and overwrite the referrals set automatically by the server. The ability to control referrals helps you to optimize security and performance by enabling you to:

- point referrals to secure ports only,
- point to a Directory Proxy Server for load balancing,
- redirect to local servers only in the case of servers separated by a WAN,
- limit referrals to a subset of masters in four-way multi-master topologies.

  For information about configuring referrals see "Setting Referrals" in Chapter 2 of the *Directory Server Administration Guide*.

Change Log

Every server acting as a supplier (a master replica or a hub replica,) maintains a *change log*. A change log is a record that describes the modifications that have occurred on a master replica. The supplier replays these modifications to its consumers.

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

In earlier versions of Directory Server, the change log was accessible over LDAP. Now, however, it is intended only for internal use by the server, and is stored in its own database which means that it is no longer accessible over LDAP. If you have applications that need to read the change log, you must use the retro change log plug-in for backward compatibility. For more information about the retro change log plug-in refer to “Using the Retro Change Log Plug-In” in the *Directory Server Administration Guide*.

NOTE Once entries are purged from the change log, they can no longer be replicated. You must therefore consider the number and size of the changes you expect, and provide sufficient disk space for change log. For more information, see “Multi-Master Replication Change Logging” in Chapter 5 of the *Directory Server Performance Tuning Guide*.

Replication Authentication

The consumer server authenticates the supplier server when the supplier binds to the consumer to send replication updates. This authentication process requires that the entry used by the supplier to bind to the consumer is stored on the consumer server. This entry is called the Replication Manager entry. When, in the context of replication, Directory Server Console refers to the DN or bind DN, it is referring to the DN of the Replication Manager entry.

The Replication Manager, or any entry you create to fulfill that role, must meet the following criteria:

- At least one must exist on every consumer server (whether it be a dedicated consumer, a hub, or a master in a multi-master environment.)
- This entry must not be part of the replicated data, for initialization and security reasons.

The Replication Manager entry has a special user profile that bypasses all access control rules defined on the consumer server. This special user profile is only valid in the context of replication.

When you configure replication between two servers, you must identify the Replication Manager entry on both servers:

- On the consumer server, you must specify this entry as the one authorized to perform replication updates, when you configure the consumer replicas, hub replicas, or master replicas (in the case of multi-master replication). If you use the console, the Replication Manager entry is used by default.
- On supplier server (all master and hub replicas), you must specify the bind DN of this entry when you configure the replication agreement.

The Replication Manager entry is created by default when you configure replication through Directory Server Console. You can also create your own Replication Manager entry.

If you are using SSL with replication, there are two possible methods of authentication:

- When using SSL Server Authentication, you must have a Replication Manager entry, and its associated password, in the server you are authenticating to.
- When using SSL Client Authentication you must have an entry containing a certificate in the server you are authenticating to. This entry may or may not be mapped to the Replication Manager entry.

Replication Agreement

Directory Server uses *replication agreements* to define how replication occurs between two servers. A replication agreement describes replication between *one* supplier and *one* consumer. The replication agreement is configured on the supplier, and must be *enabled* for replication to work. You can enable or disable existing replication agreements. This can be useful if you currently have no need for a replication agreement, but want to maintain its configuration for future use.

A replication agreement identifies:

- The suffix to replicate.
- The consumer server to which the data is pushed.
- The times during which replication can occur.
- The bind DN and credentials the supplier must use to bind to the consumer (see “Replication Authentication” on page 109.)
- How the connection is secured (SSL, client authentication).
- If *fractional replication* is configured, a pointer to the set of attributes to be excluded or included (see “Fractional Replication” on page 124.)

- The group and window sizes to configure the number of changes you can group into one request and the number of requests that can be sent before consumer acknowledgement is required.
- Information about the replication status for this particular agreement.
- On Solaris and Linux systems, the The level of compression used in replication.

Consumer Initialization

Consumer initialization, or total update, is the process by which all data is physically copied from the supplier to the consumer. Once you have created a replication agreement, the consumer defined by that agreement must be initialized. When a consumer has been initialized, the supplier can begin replaying, or replicating update operations to the consumer. Under normal circumstances, the consumer should not require further initialization. However, if the data on a supplier is restored from a backup, you may need to reinitialize the consumers dependent on that supplier. For example if a restored supplier is the only supplier for a consumer in the topology, consumer reinitialization may be necessary.

You can initialize consumers online or offline. For more information on the consumer initialization process see “Initializing Replicas,” in Chapter 8 of the *Directory Server Administration Guide*.

In a multi-master replication topology, the default behavior of a read-write replica that has been reinitialized from a backup or LDIF file, is to REFUSE client update requests. By default the replica remains in read-only mode indefinitely and refers any update operations to other suppliers in the topology. In this case, you must configure the replica to begin accepting updates again. See “Convergence After Multi-Master Initialization,” in Chapter 8 of the *Directory Server Administration Guide*.

Directory Server provides an advanced *binary copy* feature that can be used to clone master or consumer replicas using the binary backup files of one server to restore the identical directory contents on another server. Certain restrictions on this feature make it practical and time-efficient only for replicas with large database files. For information on the binary copy procedure and a list of the feature’s limitations see “Initializing a Replica Using Binary Copy,” in Chapter 8 of the *Directory Server Administration Guide*.

Incremental Updates

Once a consumer has been initialized, replication updates are sent to the consumer as the modifications are made on the supplier. These updates are called incremental updates. A consumer can be incrementally updated by several suppliers at once, provided that the updates originate from different replica IDs.

Data Consistency

Consistency refers to how closely the contents of replicated databases match each other at any given time. When you set up replication between two servers, part of the configuration is to schedule updates. The supplier determines when consumers must be updated, and initiates replication. Replication can start only after consumers have been initialized.

Directory Server provides the option of keeping replicas always synchronized, or of scheduling updates for a particular time of day, or day of the week. The advantage of keeping replicas always in sync is that data remains consistent across your topology. The cost, however, is the network traffic resulting from the frequent update operations. This solution is preferable when:

- 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 add and modify operations.

If you can afford to have looser data consistency, you can choose a frequency of updates that lowers the effect on network traffic. This solution is preferable when:

- 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 add and modify operations.
- You need to reduce 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 keep replicas in sync, because:

- There is a latency in the propagation of replication updates between masters.
- The master that serviced the add or modify operation does not wait for the second master to validate it before returning an “operation successful” message to the client.

Common Replication Configurations

Your replication topology determines how updates flow from server to server, and how the servers interact when propagating updates. There are five basic replication configurations, which can be combined to suit your deployment.

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

The following sections describe these configurations and provide strategies for deciding which method is most suited to your deployment.

NOTE Whatever replication configuration you implement, you must consider the schema replication. See "Schema Replication" on page 138 for more information.

Single Master Replication

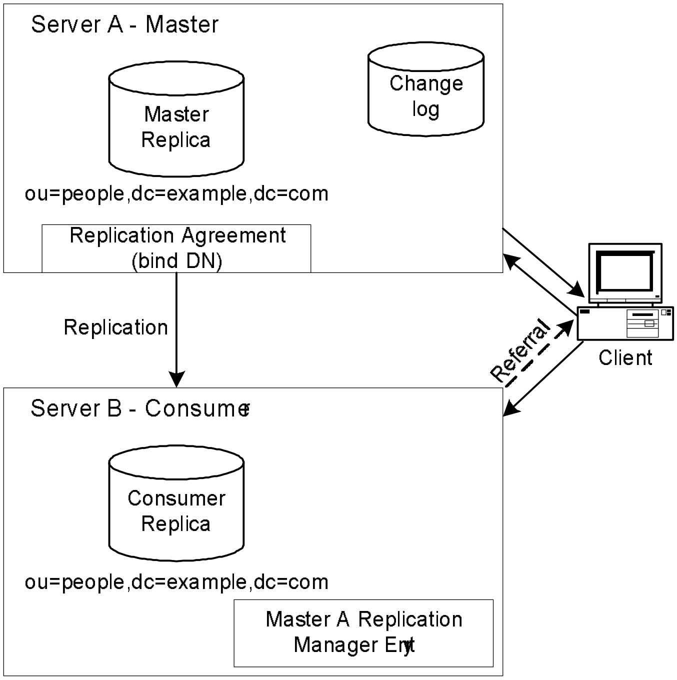
In the most basic replication configuration, a supplier copies a master replica directly to one or more consumers. In this configuration, all directory modifications are made to the master replica, and the consumers contain read-only copies of the data.

The supplier maintains a change log that records all changes made to the replica. The supplier also defines the replication agreement.

The consumer stores the entry corresponding to the Replication Manager entry, so that the consumer can authenticate the supplier when the supplier binds to send replication updates.

The supplier propagates all modifications to the consumer replicas, in accordance with the replication agreement. This basic scenario is illustrated in Figure 6-1 on page 114.

Figure 6-1 Single-Master Replication


Server A - Master
Master Replica
Change log
ou=people,dc=example,dc=com
Replication Agreement (bind DN)
Replication
Referral
Client
Server B - Consumer
Consumer Replica
ou=people,dc=example,dc=com
Master A Replication Manager Entry

In this example, the `ou=people,dc=example,dc=com` suffix receives a large number of search and update requests from clients. To distribute the load, this suffix, which is mastered on Server A, is replicated to a consumer replica located on Server B.

Server B can process and respond to search requests from clients, but cannot process requests to modify directory entries. Server B processes modification requests received from clients by sending a referral to Server A back to the client. The consumer stores referral information about the supplier, but does not forward modification requests from clients to the supplier. Instead, the client follows the referral sent back by the consumer.

Although this example shows just one server acting as a consumer, a supplier can replicate to several consumers. The total number of consumers that a single supplier can manage depends on the speed of your network and the total number of entries that are modified on a daily basis.

Multi-Master Replication

In a multi-master replication configuration, master replicas of the same data exist on more than one server. This section includes the following topics:

- Multi-Master Replication Basic Concepts
- Multi-Master Replication Capabilities
- Multi-Master Replication over Wide Area Networks
- Fully Meshed Multi-Master Topology

Multi-Master Replication Basic Concepts

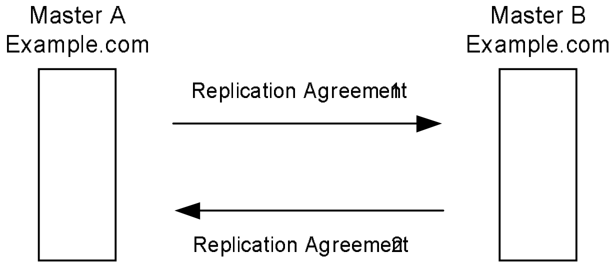
In a multi-master configuration, data can be updated simultaneously in different locations. Each master maintains a change log for its replica, and the changes that occur on each master are replicated to the other servers. This means that each master plays the role of supplier and consumer. Multi-master configurations have the following advantages:

- Automatic write failover when one supplier is inaccessible.
- Updates can be made on a local supplier in a geographically distributed environment.

When updates are sent between the two servers, the conflicting changes need to be resolved. Mostly, resolution occurs automatically, based on the timestamp associated with each change. The most recent change takes precedence. However, there are some cases where change conflicts require manual intervention in order to reach a resolution. For more information, see “Solving Common Replication Conflicts,” in Chapter 8 of the *Directory Server Administration Guide*.

Although two separate servers can have master copies of the same data, within the scope of a single replication agreement, there is only ever one supplier and one consumer. Therefore, to create a multi-master environment between two suppliers that share responsibility for the same data, you must create two replication agreements, one on each supplier. Figure 6-2 on page 116 shows this configuration:

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



In the preceding figure, Master A and Master B each hold a master replica of the same data and there are two replication agreements governing the replication flow. Master A acts as a master in the scope of Replication Agreement 1, and as a consumer in the scope of Replication Agreement 2.

Directory Server 5.2 supports a maximum of four masters in a multi-master replication topology. The number of consumer replicas and hubs is theoretically unlimited, although the number of consumers to which a single supplier can replicate will depend on the capacity of the supplier server.

Multi-Master Replication Capabilities

Directory Server 5.2 introduces a more streamlined, flexible replication protocol. This protocol enables you to:

- Replicate updates based on the replica ID. Replica ID-based updates result in improved performance because they make it possible for a consumer to be updated by multiple suppliers at the same time (provided that the updates originate from different replica IDs).
- Enable or disable a replication agreement, providing greater replication configuration flexibility. Replication agreements can be configured but left disabled, then enabled rapidly should they be required.

Multi-Master Replication over Wide Area Networks

Support for multi-master replication over Wide Area Networks (WANs), introduced in Directory Server 5.2, enables multi-master replication configurations across geographical boundaries in international, multiple data center deployments. (Prior to Directory Server 5.2, multiple masters had to be connected via high-speed, low-latency networks with minimum connection speeds of 100Mb/second, for full support, which ruled out the possibility of multi-master replication over WAN.)

From Directory Server 5.2 onwards, the replication protocol provides full asynchronous support, window and grouping mechanisms, and support for compression on Solaris and Linux systems. These features render multi-master replication over WAN a viable deployment possibility. Although the viability of multi-master replication over WAN is a direct result of these protocol improvements, they are equally valid for Local Area Network (LAN) deployments.

Due to differences in protocols, multi-master replication over WAN is not backward compatible with versions of Directory Server prior to 5.2. As a result, in a multi-master replication over WAN configuration, *all* Directory Server instances separated by a WAN *must* be 5.2 instances.

Group and Window Mechanisms

To optimize replication flow, Directory Server enables you to group changes, rather than sending them individually. It also allows you to specify a certain number of requests that can be sent to the consumer without the supplier having to wait for an acknowledgement from the consumer before continuing.

Since both the group and window mechanisms are based on entry size, optimizing replication performance using these mechanisms may be impractical if the size of your entries varies considerably. If the size of your entries is relatively constant, you can use the group and window mechanisms to optimize incremental and total updates. Note that the performance of multi-master replication over WAN will depend on the latency and bandwidth of your WAN.

For more information on adjusting the window and group size, see “Configuring Network Parameters,” in Chapter 8 of the *Directory Server Administration Guide*.

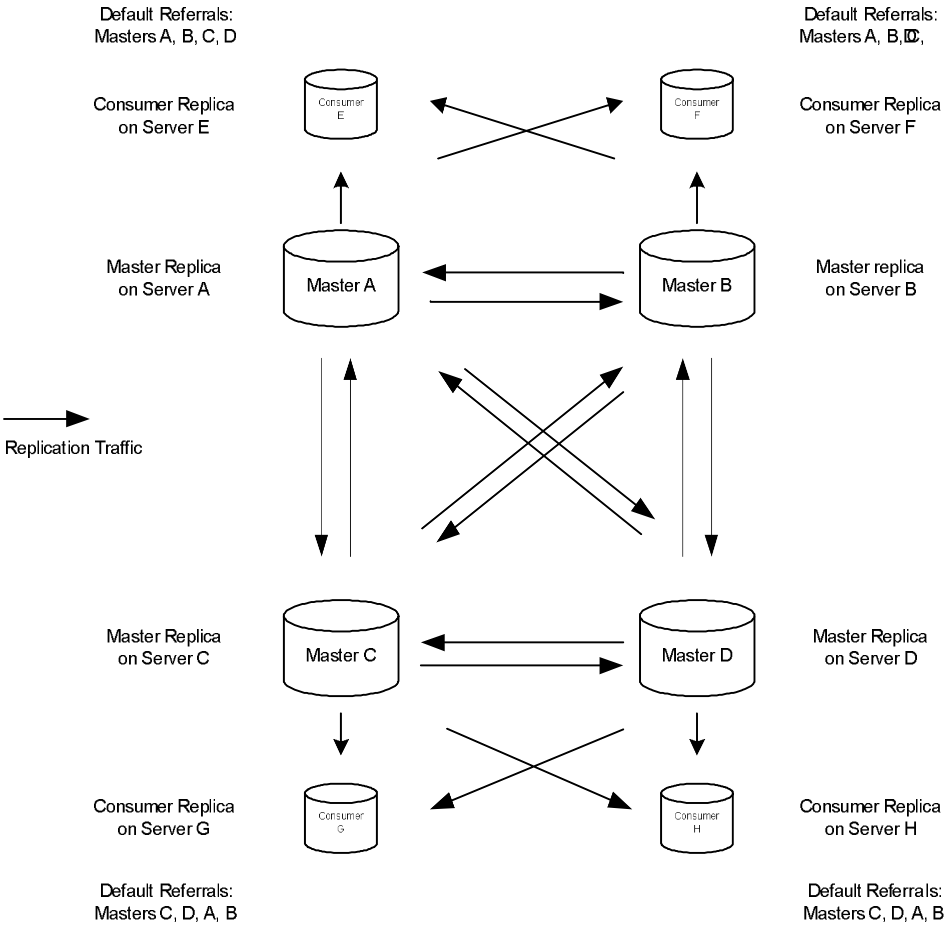
Replication Compression

In addition to the grouping and window mechanisms, Directory Server provides a compression mechanism on Solaris and Linux systems. Prior to Directory Server 5.2, limited bandwidth often caused a bottleneck in replication over WAN. Replication compression helps to streamline replication flow and avoid this bottleneck. For information on how to configure replication compression via the command line, see “`ds5ReplicaTransportCompressionLevel`,” in Chapter 4 of the *Directory Server Administration Reference*.

Fully Meshed Multi-Master Topology

A *fully meshed* topology implies that each of the masters in a topology is connected to each of the other masters. Such a topology provides high availability and guaranteed data integrity. Figure 6-3 on page 118 shows a fully meshed, four-way, multi-master topology.

Figure 6-3 Fully Meshed Four-Way Multi-Master Replication Configuration on Suffix "ou=people,dc=example,dc=com"



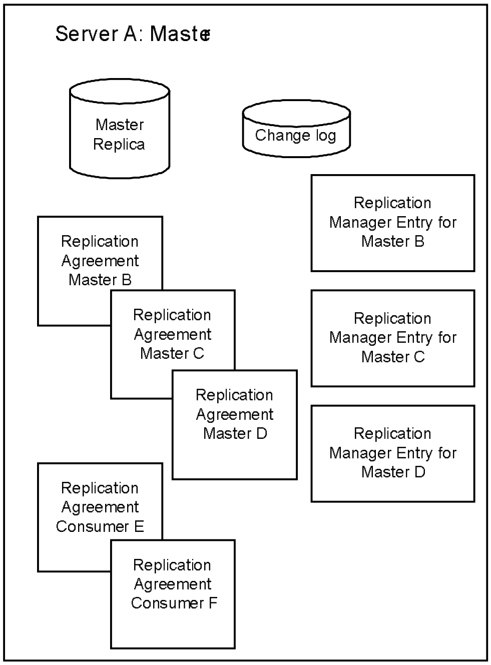
In this example, the `ou=people,dc=example,dc=com` suffix is held on four masters to ensure that it is always available for modification requests. Each master maintains its own change log. When one of the masters processes a modification request from a client, it records the operation in its change log. It then sends the replication update to the other masters, and in turn to the other consumers. This requires that the masters have replication agreements with each other, as well as with the consumers. Each master also stores a Replication Manager entry that it uses to authenticate the other masters when they bind to send replication updates.

Each consumer stores one or more entries, corresponding to the Replication Manager entries, so that they can authenticate the masters when they bind to send replication updates. It is possible for each consumer to have just one Replication Manager entry, enabling all masters to use the same Replication Manager entry for authentication. The consumers have referrals set up by default for all masters in the topology. When consumers receive modification requests from the clients, referrals to the masters are sent back to the clients by the consumers. For more information on referrals, see "Referrals" on page 108.

Although this topology is the most secure in terms of read-write failover capability, utilising this capability may impact performance. A fully meshed topology is preferable if high availability is crucial to your deployment. If your high availability requirements are not as important, or if you want to reduce replication traffic for performance reasons, you may want to opt for a "lighter" deployment in terms of read-write failover.

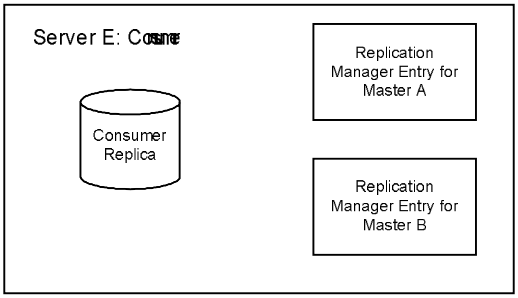
To assist you in understanding the replication elements required to configure this fully meshed, four-way, multi-master replication topology, Figure 6-4 presents a detailed view of the replication agreements, change logs, and Replication Manager entries that must be set up on master A. Figure 6-5 provides the same detailed view for consumer E.

Figure 6-4 Replication Configuration for Master A (Fully Meshed Topology)


Server A: Maste
Master Replica
Change log
Replication Manager Entry for Master B
Replication Agreement Master B
Replication Agreement Master C
Replication Manager Entry for Master C
Replication Agreement Master D
Replication Manager Entry for Master D
Replication Agreement Consumer E
Replication Agreement Consumer F

As Figure 6-4 illustrates, Master A requires a master replica, a change log and Replication Manager entries for Masters B, C, and D (if you do not use the same Replication Manager entry for all four masters). Master A also requires replication agreements for Masters B, C, and D, and for consumers E and F.

Figure 6-5 Replication Configuration for Consumer Server E (Fully Meshed Topology)


Server E: Consumer
Consumer Replica
Replication Manager Entry for Master A
Replication Manager Entry for Master B

The replication configuration presented in Figure 6-5 illustrates that Consumer E requires a consumer replica and Replication Manager entries to authenticate Master A and Master B when they bind to send replication updates.

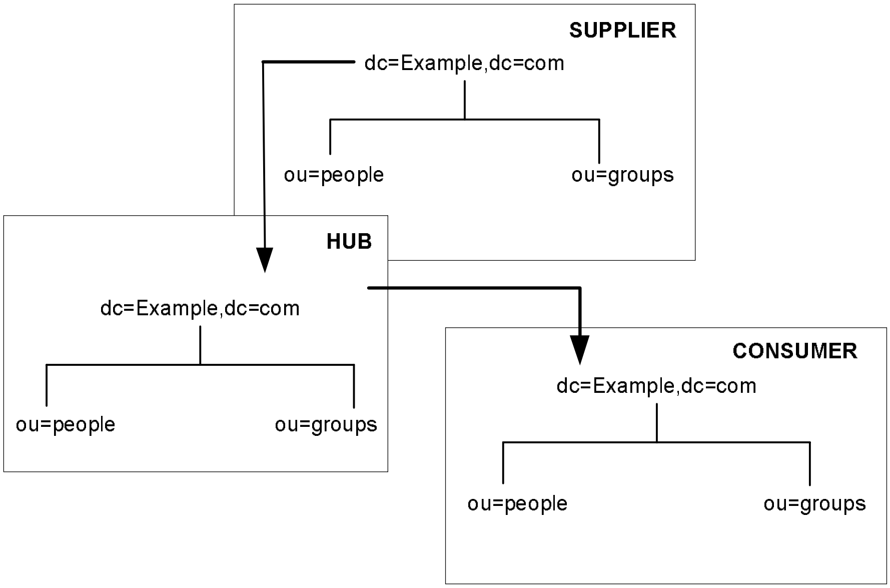
Cascading Replication

In a cascading replication configuration, a server acting as a *hub* receives updates from a server acting as a supplier, and replays those updates to consumers. The hub is a hybrid: it holds a read-only copy of the data, like a consumer *and* it maintains a change log like a supplier.

Hubs pass on copies of the master data as they are received from the original masters and refer update requests from directory clients to masters.

Figure 6-6 illustrates a cascading replication configuration:

Figure 6-6 Cascading Replication Configuration

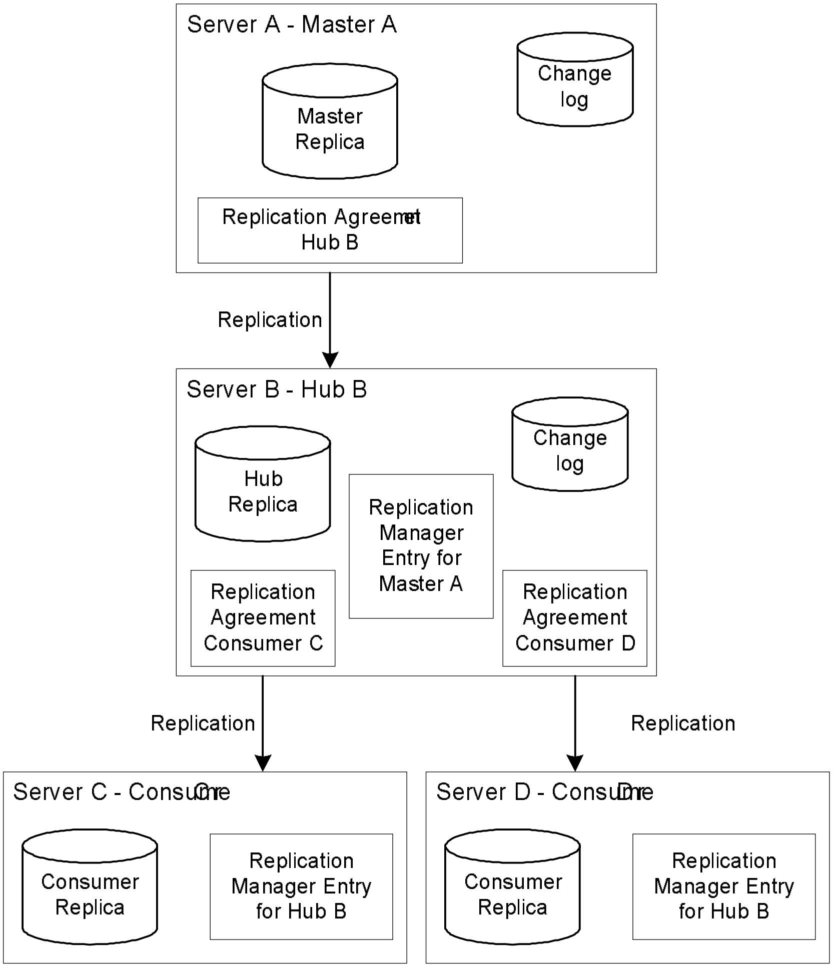


Cascading replication is particularly useful in the following cases:

- When you need to balance heavy traffic loads. Because the masters in a replication topology handle all update traffic, it may put them under a heavy load to support replication traffic to consumers as well. You can off-load replication traffic to a hub that can service replication updates to a large number of consumers.
- To reduce connection costs by using a local hub in geographically distributed environments.
- To increase performance of your directory service: if you direct all client applications performing read operations to the consumers, and all those performing update operations to the master, you can remove all of the indexes (except system indexes) from your hub. This will increase the speed of replication between the master and the hub.

Figure 6-7 shows how the servers described in the previous example are configured in terms of Replication Agreements, change logs, and default referrals.

Figure 6-7 Server Configuration in Cascading Replication


Server A - Master A
Master Replica
Change log
Replication Agreement Hub B
Replication
Server B - Hub B
Hub Replica
Change log
Replication Manager Entry for Master A
Replication Agreement Consumer C
Replication Agreement Consumer D
Replication
Replication
Server C - Consumer C
Consumer Replica
Replication Manager Entry for Hub B
Server D - Consumer D
Consumer Replica
Replication Manager Entry for Hub B

In this example, Hub B is used to relay replication updates to Consumers C and D, leaving Master A with more resources to process directory updates. The master and the hub both maintain a change log. However, only the master can process directory modification requests from clients. The hub contains a Replication Manager entry for Master A, so that Master A can bind to the hub to send replication updates. Consumers C and D both contain Replication Manager entries for Hub B, which it uses to authenticate when sending its updates to the consumers.

The consumers and the hub can process search requests received from clients, but in the case of modification requests, send the client a referral to the master. Figure 6-7 shows that Consumer C and D have a referral to Master A. These are the automatic referrals that are created when you create the replication agreement between the hub and the consumers. You can, however, overwrite these referrals for performance or security reasons. For more information see the “Referrals” on page 108.

Mixed Environments

You can combine any of the replication configurations outlined in the previous sections to suit your deployment. For example, you could combine a multi-master configuration with a cascading configuration to produce a topology similar to that illustrated in Figure 6-8:

Figure 6-8 Combined Multi-Master and Cascading Replication

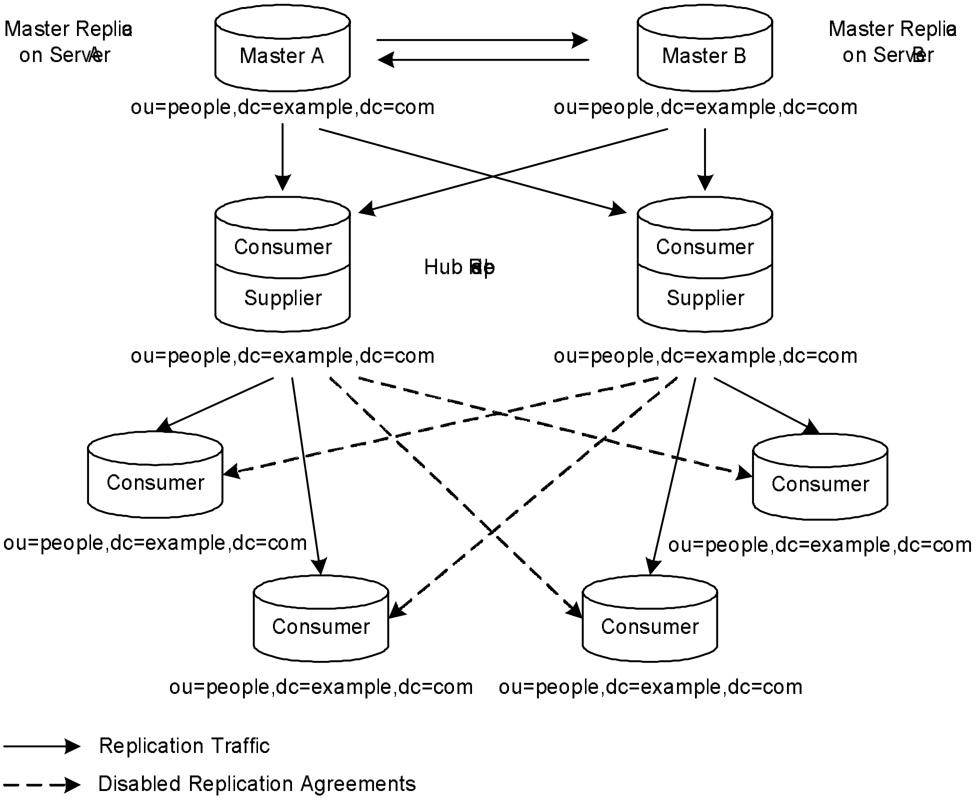

Master Replica
on Server A
Master A
Master B
Master Replica
on Server B
ou=people,dc=example,dc=com
ou=people,dc=example,dc=com
Consumer
Supplier
Hub Replica
Consumer
Supplier
ou=people,dc=example,dc=com
ou=people,dc=example,dc=com
Consumer
Consumer
ou=people,dc=example,dc=com
ou=people,dc=example,dc=com
Consumer
Consumer
ou=people,dc=example,dc=com
ou=people,dc=example,dc=com
Replication Traffic
Disabled Replication Agreements

Figure 6-8 shows two masters and two hubs replicating data to four consumers. As in the previous scenario, the hubs are used to balance the load of replication updates by sharing it between the masters and the hubs.

In this example, the dotted lines represent disabled replication agreements. If these replication agreements are not enabled, the topology presented contains a single point of failure (if one of the hubs were to go off line.) In deploying this topology, you would need to weigh up performance requirements against high availability requirements to determine whether you enable all replication agreements to provide full read-write failover.

Fractional Replication

While the unit of replication is the suffix or subsuffix, fractional replication functionality provides a greater degree of granularity in replication. Fractional replication enables you to replicate a subset of the attributes of all entries in a suffix or subsuffix.

Benefits of Fractional Replication

Fractional replication is useful in a variety of scenarios.

When you need to synchronize between intranet and extranet servers and filter out content for security reasons, fractional replication provides the filtering functionality.

Because fractional replication enables you to be selective in what you replicate, you can reduce replication costs. If your deployment requires only certain attributes to be available everywhere, you can use the fractional replication functionality to replicate the required attributes only, rather than replicating all attributes.

For example, you may want e-mail and phone attributes to be replicated but not all attributes on a user entry, particularly if the other attributes are modified frequently and generate heavy network traffic. Fractional replication enables you to filter in the required attributes and reduce traffic to a minimum. This filtering functionality is particularly valuable where replication is over a WAN.

Fractional replication is *not* backward compatible with versions of Directory Server prior to 5.2. If you are using fractional replication, you *must* ensure that *all* other instances of Directory Server are 5.2 instances.

Configuring Fractional Replication

Fractional replication can be configured easily from Directory Server Console. Configuring fractional replication involves either:

- Specifying the list of attributes that will be *included* in the replication.
- Selecting all attributes and specifying those that will be *excluded*.

Under most circumstances, an exclusion configuration approach is preferable. The complexity of certain features such as ACIs, CoS and Roles, and the dependency these features have on certain attributes, make managing a list of excluded attributes far safer, and less prone to human error, than managing a list of included attributes.

When configuring fractional replication, the server being replicated to must be a read-only replica.

Generally, you replicate all required attributes for each entry as defined in the schema, to avoid schema violations. If you want to filter out certain required attributes using fractional replication, you must disable schema checking. If schema checking is enabled with fractional replication, you may not be able to initialize the server off line (from an LDIF file). This is because the server will not allow you to load the LDIF file if required attributes are filtered out. If you have disabled schema checking on a fractional consumer replica, the whole server instance on which that fractional consumer replica resides will not enforce the schema. Because schema is pushed by suppliers in fractional replication configurations, the schema on the fractional consumer replica will be a copy of the master replica’s schema. Therefore, it will not correspond to the fractional replication configuration being applied.

Before modifying a fractional replication configuration, you must disable the replication agreements it affects. Once you have modified the configuration, you will need to re-enable the replication agreements and re-initialize the consumers so that the new configuration is taken into account.

For more information see “Configuring Fractional Replication,” in Chapter 8 of the *Directory Server Administration Guide*.

Defining a Replication Strategy

Your replication strategy will be determined by the service you want to provide. This section provides replication topology examples that focus on the following:

- Using Replication for High Availability
- Using Replication for Local Availability
- Using Replication for Load Balancing

To assess how important each of these aspects is in your deployment, start by performing a survey of your network, users, client applications, and how they will use the directory service. For guidelines on performing this survey, refer to "Performing a Replication Survey" on page 126.

When you understand your replication strategy, you can start deploying Directory Server. Putting your directory into production in stages, will give you a better sense of the load that your enterprise places on the directory. Unless you can base your load analysis on an operating directory, be prepared to alter your directory as you develop a better understanding of how the directory is used.

The following sections describe the main factors affecting your replication strategy:

- Performing a Replication Survey
- Replication Resource Requirements
- Replication Backward Compatibility
- Using Replication for High Availability
- Using Replication for Local Availability
- Using Replication for Load Balancing
- Example Replication Strategy for a Small Site
- Example Replication Strategy for a Large Site

Performing a Replication Survey

When performing a replication survey, concentrate on gathering the following information:

- Quality of the networks connecting different buildings or remote sites, and the amount of available bandwidth.

- Physical location of users, number of users at each site, and potential user activity on the directory.

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

- 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 e-mail message it handles. Other products that rely on the directory are typically products such as authentication applications, or meta-directory applications. For each one you must determine the type and frequency of operations performed on the directory.

- Approximate number and size of entries stored in the directory.

Replication Resource Requirements

Replication functionality requires system resources. Consider the following resource requirements when defining your replication strategy:

- Disk usage

  On suppliers, the change log is written to after each update operation. If a supplier contains multiple replicated suffixes, the change log will be updated more frequently, and disk usage will be higher.

  Consumers must be at least equivalent to suppliers in terms of machine size, to prevent bottlenecks.

- Server threads

  Each replication agreement creates two additional threads. Replication agreement threads are separate from operational threads. If there are several replication agreements, 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).

Replication Backward Compatibility

If you are using multiple Directory Server versions in a replication topology you should take into account the backward compatibility information in Table 6-1 on page 128, to ensure that replication functions correctly. This table presents the supplier and consumer combinations that are possible between different versions of Directory Server.

Table 6-1 Replication Backward Compatibility With Different Directory Server Versions

	4.x Consumer	**5.0/5.1 Consumer**	**5.0/5.1 Master**	**5.2 Consumer**	**5.2 Master**	**5.x Hub Supplier**
4.x Master	Yes	Yes	Yes	Yes	Yes	No
5.0/5.1 Master	No	Yes	Yes	Yes	Yes	Yes
5.2 Master	No	Yes	Yes	Yes	Yes	Yes

If you are using replication with different Directory Server versions, take note of the following:

- If you configure a 4.x master to replicate to a 5.x master and you enable *legacy replication* on the 5.x master, the 5.x master will not be able to receive either client updates or replication updates from other 5.x masters in the topology. It will only receive replication updates from the 4.x master. However, when legacy replication is disabled, the 5.x master will resume fully-operational master replication behavior.
- When you are replicating from a 5.2 server to a 5.0/5.1 server, the features and enhancements that are new in 5.2 should not be used. These features include fractional replication, multiple password policies, multi-master replication over WAN, and online promotion and demotion.
- The `nsslapd-schema-replicate-useronly` attribute must be set to `on` to make sure that 5.1 servers are not disrupted by 5.2 schema extensions.

Using Replication for High Availability

Replication can be used 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 requires more backup servers.

NOTE You should not use replication as a replacement for a regular data backup policy. For information on backing up directory data, refer to "Choosing a Backup Method" on page 199 and to "Backing Up Data" in the *Directory Server Administration Guide.*

To guarantee write failover for directory clients, you should use a multi-master replication topology. If read failover is sufficient, and your directory is not geographically dispersed, you can use single-master replication.

LDAP client applications are usually configured to search one LDAP server only. 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 Directory Server. Therefore, you may need to use either DNS round robins or network sorts to provide failover to your backup Directory Servers. For information on setting up and using DNS round robins or network sorts, see your DNS documentation.

To maintain write failover over two geographically distributed sites, you can use four-way multi-master replication over WAN. In this scenario, you would set up two master servers in one location and two master servers in the second location, and configure them to be fully meshed over the WAN. This safeguards against the eventuality of one master going off line.

Alternatively, you can use the Sun Java System Directory Proxy Server product. For more information on Directory Proxy Server, see `http://www.sun.com/software/products/directory_proxy/home_dir_proxy.html`.

Using Replication for Local Availability

You can use replication for local availability when:

- You need a local master copy of the data.

  This is important for large, multinational enterprises that need to maintain directory information of interest only to users in a specific geographical location. Having a local master copy of the data is also useful in enterprises where there is a need for data to be managed at a divisional or organizational level.

- You are using unreliable or intermittent network connections.

  Having a copy of data locally means that your directory will still be available should a network problem occur.

- Your networks periodically experience extremely heavy loads that may cause the performance of your directory to be reduced.

  Enterprises with ageing networks may experience these conditions during normal business hours.

- You want to reduce both the network load and the work load on the master replica.

  Your network may be perfectly reliable and available, but you want to reduce network costs.

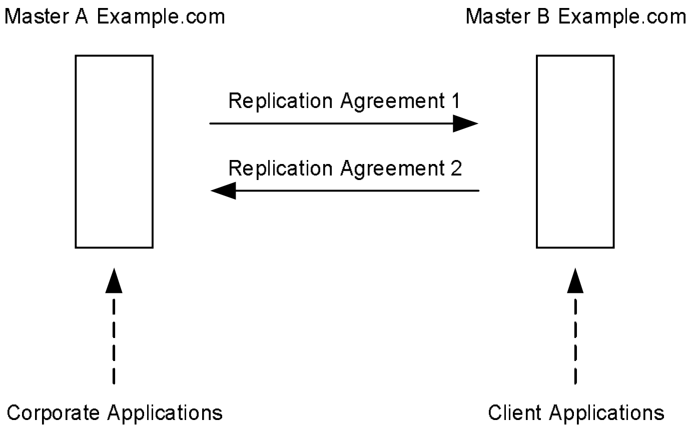
Using Replication for Load Balancing

Replication can balance the load on your Directory Server 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 server containing the master replica).
- By dedicating special servers to specific tasks, such as supporting mail server activities.

Figure 6-9 shows how replication can be used to divide directory activities between different types of applications, thereby reducing the load placed on each supplier server.

Figure 6-9 Using Multi-Master Replication for Load Balancing



Replicating directory data also balances the load placed on your network. Where possible, you should move data to servers that can be accessed using a fast and reliable network connection.

Directory entries generally average around one KB in size. Therefore, an entire entry lookup adds about one KB to your network load each time. 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. If you have a slow, heavily loaded, or unreliable WAN, you may need to replicate your directory tree to a local server.

Note that the benefit of locally available data must be weighed up against the cost of the increased network traffic caused by replication. For example, if you replicate 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 changes frequently, yet you have only a few users at the remote site performing a few directory lookups per day.

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, you could be increasing your network load by 100 MB per day. However, if your remote site has only 100 employees, and they are performing an average of ten directory lookups per day, the network load caused by their directory access is only one MB per day.

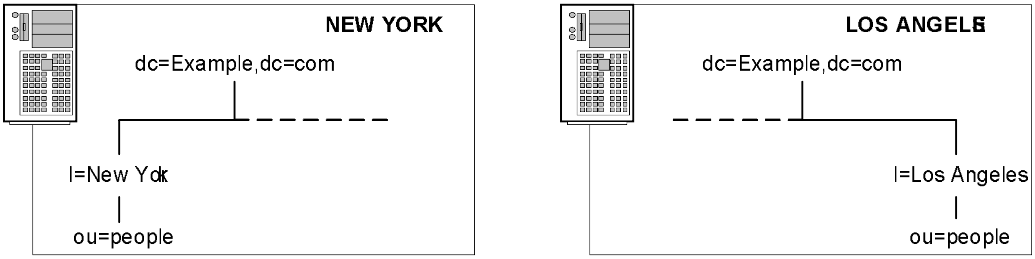
Given the difference in network load caused by replication versus that caused by normal directory usage, you may decide that replication purely for network load-balancing is not viable. On the other hand, you may find that the benefits of locally available directory data far outweigh any considerations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

A 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 112.

Example of Network Load Balancing

Suppose your enterprise has offices in two cities. Each office manages a separate subtree, as illustrated in Figure 6-10:

Figure 6-10 New York and Los Angeles Subtrees in Respective Geographical Locations



Each office contains a high-speed LAN, but uses a dial-up connection to network between the two cities. To balance network load:

- Select one server in each office to be the master for the locally managed data.

  Replicate locally managed data from that server to the corresponding master in the remote office. Having a master copy of the data in each location prevents users from having to perform update and lookup operations over the dial-up connection, which optimizes performance.

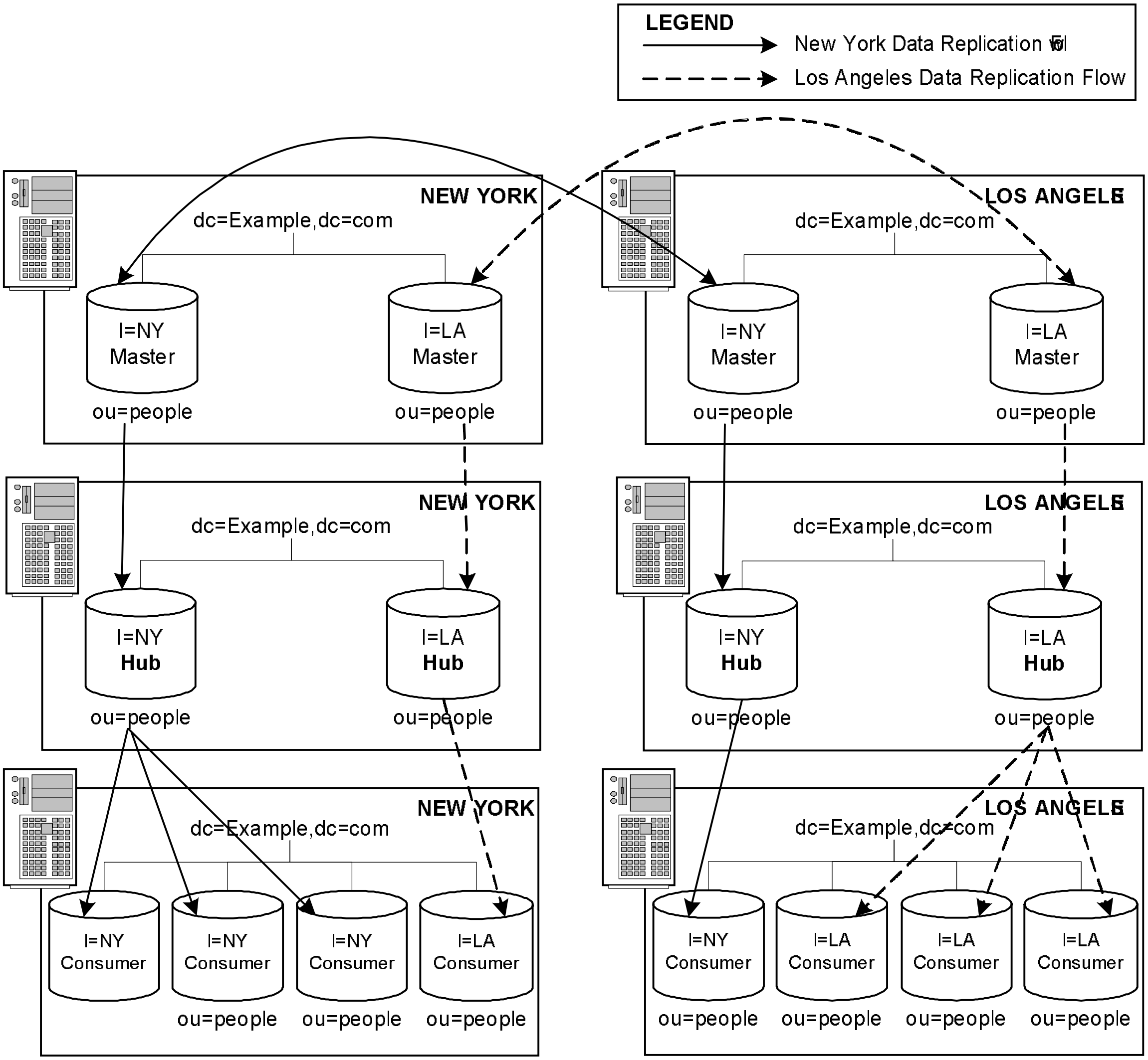
- Replicate the directory tree on each master (including data supplied from the remote office) to at least one local Directory Server to ensure local availability of directory data.

- Configure cascading replication in each location with an increased number of consumers dedicated to lookups on the local data to provide further load balancing.

The New York office has to deal with more New York specific lookups than Los Angeles specific lookups and as a result, our example shows the New York office with three New York data consumers and one Los Angeles consumer. Following the same logic, the Los Angeles office has three Los Angeles data consumers and one New York data consumer.

This network load balancing configuration is illustrated in Figure 6-11:

Figure 6-11 Load Balancing Using Multi-Master and Cascading Replication



Example of Load Balancing for Improved Performance

In this example, the directory contains 15,000,000 entries, is accessed by 10,000,000 users, and each user performs ten directory lookups a day. The messaging server handles 250,000,000 mail messages a day, and performs five directory lookups for every mail message that it handles. There are approximately 1,250,000,000 directory lookups per day, just as a result of mail. The total combined traffic is, therefore, 1,350,000,000 directory lookups per day.

Assuming an eight-hour business day, with the directory users clustered in four time zones, the business day (or peak usage) across the four time zones is 12 hours. Therefore, the directory must support 1,350,000,000 lookups in a 12-hour day. This equates to 31,250 lookups per second (1,350,000,000 / (60*60*12)). That is:

10,000,000 users	10 lookups per user =	100,000,000 reads/day
250,000,000 messages	5 lookups per message =	1,250,000,000 reads/day
	Total reads/day =	1,350,000,000
12-hour day includes 43,200 seconds	Total reads/second =	31,250

Assume a combination of CPU and RAM that allows the directory to support 5,000 reads per second. Simple division indicates that in this scenario, you need at least six or seven Directory Servers to support this load. For enterprises with 10,000,000 directory users, you would add more Directory Servers for local availability.

A single Directory Server 5.2 with the appropriate hardware and configuration can sustain much more than the 5,000 reads per second.

In this scenario, you would 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 masters to replicate to one or more hubs.

  The read, search, and compare requests serviced by your directory should be targeted at the consumers, thereby freeing the masters to handle write requests. For more information, see “Cascading Replication” on page 120.

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

Replicating to local sites helps balance the load on your servers and your network, and ensures high availability of directory data.

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

  Use DNS sort to ensure that 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 fast (100 MB per second) and lightly used network. The network is stable and you are reasonably confident of the reliability of your server hardware and OS platforms. You are also 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 when 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 Sun Java System Directory Proxy Server. For more information on Directory Proxy Server, see `http://www.sun.com/software/products/directory_proxy/home_dir_proxy.html`.

Example Replication Strategy for a Large Site

Suppose your entire enterprise is contained within two buildings. Each building has a fast (100 MB per second) and lightly used network. The network is stable and you are reasonably confident of the reliability of your server hardware and OS platforms. You are also 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.

A typical replication strategy for this scenario would be:

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

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

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

  Use multi-master replication to ensure write-failover.

- Create two replicas in the second building (Building B).
- If there is no need for close consistency between the master copy of the data and the replicated copies, schedule replication so that it occurs only during off peak hours.

Replication Strategy for a Large, International Enterprise

Suppose your enterprise comprises two major data centers - one in France and the other in the USA - separated by a WAN. Not only do you need to replicate over a WAN, but you do not want your partners to have access to all data and want to filter out certain data. Your network is very busy during normal business hours.

A typical replication strategy for this scenario would be:

- Hold master copies of directory data on servers in both data centers.
- For write-failover within the French and American sites, replicate your data to a second master in each data center.
- Deploy a fully meshed, four-way, multi-master replication topology between France and the USA to provide high-availability and write-failover across the deployment.
- Deploy as many consumers as you require in each data center to reduce the load on your masters in terms of directory lookups.
- Set up fractional replication agreements between masters and consumers in both geographical locations, to filter out the data you do not wish your partners to access.
- Schedule replication so that it occurs only during off peak hours to optimize bandwidth.

Using Replication with Other Directory Features

Replication interacts with other Directory Server features to provide advanced replication functionality. The following sections describe feature interactions to assist you in designing 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, "Access Control, Authentication, and Encryption."

Replication and Directory Server Plug-Ins

You can use replication with most of the plug-ins delivered with Directory Server. There are some exceptions and limitations, which are described in the following sections:

- Replication and the Retro Change Log Plug-In
- Replication and the Referential Integrity Plug-In
- Replication and Pre-Operation and Post-Operation Plug-Ins

Replication and the Retro Change Log Plug-In

The retro change log plug-in provides backward compatibility with 4.x releases of Directory Server. This plug-in does not log the result of conflict resolution (that is, the changes actually applied on the entry.) By default the retro change log plug-in is disabled, and must be enabled using Directory Server Console or the command line.

The retro change log plug-in was not designed to function in a multi-master replication environment. If the plug-in is configured on two servers (in a multi-master environment), there is no guarantee that the changes will be logged in the same order, with the same change numbers. If the order of changes is fundamental to your application, the application cannot use two different retro change logs.

If you configure the retro change log plug-in on only one of the masters in a multi-master replication environment, the ordering problem is alleviated. However, this configuration introduces a single point of failure, should that master fail.

For more information see "Using the Retro Change Log Plug-In," in Chapter 8 of the *Directory Server Administration Guide*.

Replication and the Referential Integrity Plug-In

You can use the referential integrity plug-in with multi-master replication, provided that the plug-in is enabled on all master replicas. By default the referential integrity plug-in is disabled, and must be enabled using Directory Server Console or the command line.

Before enabling the referential integrity plug-in on servers issuing chaining requests, analyze your performance resource, time and integrity needs, as integrity checks can consume significant memory and CPU resources.

For more information, see “Using Referential Integrity with Replication,” in Chapter 2 of the *Directory Server Administration Guide*.

Replication and Pre-Operation and Post-Operation Plug-Ins

When writing pre- and post-operation plug-ins, you can specify that the plug-ins ignore any replicated operations. This is likely to be the desired plug-in behavior, in most cases. Be aware that changing replicated operations can result in unexpected behavior.

For more information, see “Pre-Operation and Post-Operation Plug-Ins, ”” in Chapter 5 of the *Directory Server Plug-In Developer’s Guide*.

Replication and Chained Suffixes

When you distribute entries using chaining, the server containing the chained suffix points to a remote server that contains the actual data. This remote server is also called a *farm server*. In this environment, you cannot replicate the chained suffix itself. You can, however, replicate the suffix that contains the actual data on the remote server. You must configure the replication agreement on the remote server and not on the server containing the chained suffix.

Do *not* use replication as a backup for chained suffixes. You must back up chained suffixes manually. For more information about chaining and entry distribution refer to “Referrals and Chaining” on page 89.

Schema Replication

When Directory Server is used in a replicated environment, the schema must be consistent across all of the servers that participate in replication. If the schema is not consistent across servers, the replication process is likely to generate errors.

The best way to guarantee schema consistency is to make schema modifications on a single master, even in a multi-master replication topology. There is no conflict resolution with regard to schema modifications. Therefore, if you make schema modifications on two masters in a multi-master topology, the master that was updated last will propagate its schema to the consumer. This means that you risk losing modifications made to one master, if different modifications are made to another master at a later stage.

Never update the schema on a consumer. If you update the schema on a consumer, and as a result the version of the schema on the supplier is older than the version on the consumer, you will encounter errors when you attempt to search the consumer or update the supplier.

Schema replication occurs automatically. If replication has been configured between a supplier and a consumer, the schema is replicated by default.

The logic used by Directory Server for schema replication can be described as follows:

1. Before pushing data to a consumer, the supplier checks whether its own version of the schema is in sync with the version of the schema held by the consumer.
2. If the schema entries on both supplier and consumer are the same, the replication operation proceeds.
3. If the version of the schema on the supplier is more recent than the version on the consumer, the supplier replicates its schema to the consumer before proceeding with data replication.

NOTE ACIs present in the schema are replicated.

Changes made to custom schema files are only replicated if the schema is updated using LDAP or Directory Server Console. Custom schema files should be copied to each server to maintain the information in the same schema file on all servers. For more information, see “Replicating Schema Definitions,” in Chapter 9 of the *Directory Server Administration Guide*.

Replicating only user-defined schema reduces the amount of data transferred and thus speeds up the replication of schema. For more information, see “Limiting Schema Replication,” in Chapter 9 of the *Directory Server Administration Guide*.

Replication and Multiple Password Policies

When using multiple password policies, you must replicate the LDAP subentry containing the definition of the policy to apply to the replicated entries. If you do not do so, the default password policy is applied. This policy will not work for entries that have been configured to use a non-default password policy.

If you replicate these entries to a 5.0/5.1 server, replication functions correctly but the password policy is not enforced on the 5.0/5.1 server. Multiple password policy functionality is supported for Directory Server 5.2 only.

Replication Monitoring

Directory Server 5.2 provides command-line tools that enable replication monitoring between servers. The ability to monitor replication activity assists in identifying the causes of replication problems. All the monitoring tools can be used over a secure connection.

The replication monitoring tools constitute an LDAP client, and as such, need to authenticate to the server and use a bind DN that has read access to `cn=config`.

The following replication monitoring tools are provided:

- `insync` - indicates the state of synchronization between a master replica and one or more consumer replicas.
- `entrycmp` - enables you to compare the same entry on two or more servers.
- `repldisc` - enables you to “discover” a replication topology. Topology discovery starts with one server and builds a graph of all known servers within the topology. This replication topology discovery tool is useful for large, complex deployments where it might be difficult to recall the global topology you have deployed.

For more information regarding the replication monitoring tools, refer to “Replication Monitoring Tools” in the *Directory Server Reference*. For information on the monitoring possibilities available with certain replication attributes, see the replication attributes section of in “Core Server Configuration Attributes” in the *Directory Server Reference*.

Chapter 7

Access Control, Authentication, and Encryption

How you secure data in Directory Server has an impact on all other areas of design. This chapter describes how to analyze your security needs and explains how to design your directory to meet those needs. It includes the following sections:

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 broad categories:

- Unauthorized Access
- Unauthorized Tampering
- Denial of Service

Unauthorized Access

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

- Unauthorized access to data via data-fetching operations
- Unauthorized access to reusable client authentication information by monitoring the access of others
- Unauthorized access to data by monitoring the access of others

For example, an unauthorized client can use another client’s credentials to access the data, 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 authentication methods, password policies, and access control mechanisms provided by Directory Server offer efficient ways of preventing unauthorized access. Refer to “Selecting Appropriate Authentication Methods” on page 146, “Designing Password Policies” on page 152, and “Designing Access Control” on page 161 for more information.

Unauthorized Tampering

If intruders gain access to your directory or intercept communication between Directory Server and a client application, they have the potential to modify (or tamper with) directory data. These unauthorized modifications may include:

- Unauthorized modification of data
- Unauthorized modification of configuration information

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.

If your directory cannot detect tampering, an attacker could alter a client's request to the server, cancel the request, or change the server's response to the client. The Secure Socket Layer (SSL) protocol and similar technologies can solve this problem by signing information at either end of the connection. For more information about using SSL with Directory Server, refer to "Securing Connections With SSL" on page 170.

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.

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, refer to "Setting Individual Resource Limits" in the *Directory Server Administration Guide*.

Overview of Security Methods

Your security policy must be strong enough to prevent sensitive information from being modified or retrieved by unauthorized users, but simple enough to administer easily. A complex security policy can lead to mistakes that either prevent people from accessing the information that you want them to access or, worse, allow people to modify or retrieve directory information that you do not want them to access.

Directory Server provides the following security methods:

- Authentication

  A means for one party to verify 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.

- Secure Sockets Layer (SSL)

  Maintains the integrity of information. If encryption and message digests are applied to the information being sent, 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.

Analyzing Your Security Needs

When you performed your site survey in Chapter 2, “Planning and Accessing Directory Data,” you made certain 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.

How 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 primarily:

- Providing users and applications with access to the information they need to perform their jobs.
- Protecting sensitive employee or company data from general access.
- Ensuring information integrity.

If your directory serves an extranet, or supports e-commerce applications over the Internet, you also need to offer your customers and business partners a guarantee of privacy.

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

- Determining Access Rights
- Ensuring Data Privacy and Integrity
- Conducting Security Audits

Determining Access Rights

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

You can grant access rights in two ways:

- Grant all categories of users the ability to perform self-administration or delegate management, 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.

Regardless of how you decide 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 146. For information about restricting access to directory information, refer to “Designing Access Control” on page 161.

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
- Using certificates to sign data
- Encrypting data transfers

For information about encryption methods provided in Directory Server, refer to "Password Storage Scheme" on page 155 and "Encrypting Attributes" on page 171. For information about signing data, refer to "Securing Connections With SSL" on page 170.

Conducting Security Audits

Security audits ensure that the implementation of a security policy is working. There are several aspects to a security audit. A basic audit procedure is to automate testing using a network-based security scanner that identifies vulnerabilities on your network. You can also conduct audits by examining the log files and the information recorded by the SNMP agents. For more information about monitoring your directory, refer to Chapter 8, "Directory Server Monitoring."

Selecting Appropriate Authentication Methods

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

Directory Server supports the following authentication mechanisms:

- Anonymous Access
- Simple Password
- Proxy Authorization
- Simple Password Over a Secure Connection
- Certificate-Based Client Authentication

- SASL-Based Client Authentication

The same authentication mechanism applies to 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 152.

Anonymous Access

Anonymous access provides the easiest form of access to your directory. It makes data available to any user of your directory, regardless of whether they have authenticated.

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 allow anonymous access only 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 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:

- Grants anonymous access if the user does not attempt to provide a password.
- Denies access if the user provides any non-null string for the password.

For example, consider the following ldapsearch command:

```
% ldapsearch -h ds.example.com -D "cn=joe,dc=Example,dc=com"
  -w secretpwd -b "cn=joe,dc=Example,dc=com" "objectclass=*"
```

```
ldap_simple_bind_s: Invalid credentials
```

Although Directory Server 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, clients must authenticate to Directory Server before they 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 Directory Server via a bind operation in which it provides a distinguished name and its 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 administrative entry rather than as a person. Directory Server 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, `bjensen`).
2. The LDAP client application searches the directory for that identifier and returns the associated distinguished name (such as `uid=bjensen,ou=people,dc=Example,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. If a rogue user is listening, this can compromise the security of your Directory Server 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.

Proxy Authorization

Proxy authorization is a special form of access control: a user that binds to Directory Server using his own identity is granted the rights of another user, through proxy authorization.

Using proxy authorization, directory administrators can request access to Directory Server by assuming the identity of a regular user. They bind to the directory using their own credentials, but for purposes of access control evaluation, are granted the rights of the regular user. This assumed identity is called the *proxy user*, and the DN of that user, the *proxy DN*.

To configure Directory Server to allow proxy requests:

- You must grant the administrators the right to proxy as other users
- You must grant your regular users normal access rights as defined in your access control policy.

NOTE You can grant proxy rights to any users of the directory except the Directory Manager. You should exercise great care when granting proxy rights because you grant the right to specify any DN (except the Directory Manager DN) as the proxy DN.

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

For more information on proxy authorization, refer to Chapter 6, “Managing Access Control” in the *Directory Server Administration Guide.*

Simple Password Over a Secure Connection

A secure connection uses encryption to make data unreadable to third parties while it is sent over the network between Directory Server and its clients. Clients may establish secure connections in either of the following ways:

- Bind to the secure port using the Secure Socket Layer (SSL).
- Bind to the insecure port with anonymous access, and then send the Start TLS control to begin using Transport Layer Security (TLS).

In either case, the server must have a security certificate, and the client must be configured to trust this certificate. The server sends its certificate to the client to perform *server authentication* using public-key cryptography. As a result, the client knows that it is connected to the intended server and that the server is not being tampered with.

The client and server then begin to encrypt all data transmitted through the connection for privacy. The client sends the bind DN and password on the encrypted connection to authenticate the user. All further operations are performed with the identity of the user or with a proxy identity if the bind DN has proxy rights to other user identities. In all cases, the results of operations are encrypted when they are returned to the client.

For more information about SSL, refer to "Securing Connections With SSL" on page 170. For information about configuring certificates and activating SSL, see Chapter 11, "Managing Authentication and Encryption" in the *Directory Server Administration Guide*.

Certificate-Based Client Authentication

When establishing encrypted connections over SSL or TLS, you can also configure the server to require *client authentication*. The client must send its credentials to the server to confirm the identity of the user. The user's credentials, and not the DN, are used to determine the bind DN. Client authentication protects against user impersonation and is the most secure type of connection.

One form of credentials that a client may send is the user's certificate. To perform certificate-based authentication, the directory must be configured to perform certificate mapping, and all users must store a copy of their certificate in their entry. After receiving a user certificate from a client, the server performs a mapping based on the certificate contents to find a user entry in the directory. This entry must contain an exact copy of the certificate for the user to be positively identified. All operations proceed using this entry's DN as the bind DN, and all results are encrypted over the SSL or TLS connection.

For more information about certificate mapping, see “Using Client Authentication” in Chapter 9 of the *Administration Server Administration Guide*, and “Configuring Certificate-Based Authentication in Clients” in Chapter 11 of the *Directory Server Administration Guide*.

Directory Server now supports a version of the Network Security Services (NSS) component with a certificate base that allows for larger certificates than were previously possible. In addition, a directory is automatically created next to the certificate database when an object larger than 14 KB is imported.

SASL-Based Client Authentication

Another type of client authentication during an SSL or TLS connection uses the Simple Authentication and Security Layer (SASL) to establish the identity of the client. Directory Server supports the following mechanisms through the generic security interface of SASL:

- DIGEST-MD5 - This mechanism authenticates clients by comparing a hashed value sent by the client with a hash of the user's password. However, because the mechanism must read user passwords, all users wishing to be authenticated through DIGEST-MD5 must have `{CLEAR}` passwords in the directory.
- GSSAPI - Available only on the Solaris Operating System, the General Security Services API (GSSAPI) allows Directory Server to interact with the Kerberos V5 security system to identify a user. The client application must present its credentials to the Kerberos system, which in turn validates the user's identity to Directory Server.

When using either SASL mechanism, the server must also be configured to perform identity mapping. The SASL credentials are called the *Principal*, and each mechanism must have specific mappings to determine the bind DN from the contents of the Principal. When the Principal is mapped to a single user entry and the SASL mechanism validates that user's identity, the user's DN is the bind DN for the connection.

For more information, see “Using SASL DIGEST-MD5 in Clients” and “Using Kerberos SASL GSSAPI in Clients” in Chapter 11 of the *Directory Server Administration Guide*.

Preventing Authentication by Account Inactivation

You can temporarily inactivate a user account or a set of accounts. Once inactivated, a user cannot bind to Directory Server, 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. (The `nsAccountLock` attribute should never be modified manually, but using the command-line utilities.)

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 "Managed, Filtered, and Nested Roles" on page 67.

Designing Password Policies

A password policy is a set of rules that govern how passwords are administered in a given system.

The password policy functionality provided in Directory Server 5.2 enables you to configure multiple password policies as opposed to one global policy for your entire directory. You can also assign password policies either to particular users or to sets of users using the CoS and Roles functionality. This provides significantly more scope when implementing password policy security measures, because you can tailor password policies to specific users or roles.

For detailed information regarding the attributes available to build password policies, see "Password Policy Attributes" and "Account Lockout Attributes" in the *Directory Server Administration Reference*. For information on configuring and managing password policies, refer to Chapter 7, "Managing User Accounts and Passwords," in the *Directory Server Administration Guide*. This section is divided into the following topics:

- Password Policy Features
- Configuring Password Policies
- Preventing Dictionary-Style Attacks
- Password Policies in a Replicated Environment

Password Policy Features

This section describes the main password policy features, and is divided into the following sub-sections:

- User-Defined Passwords
- Password Change at First Login or Reset
- Password Expiration
- Expiration Warning
- Password Syntax Checking
- Password Length
- Password Minimum Age
- Password History
- Password Storage Scheme

User-Defined Passwords

You can set up your password policy to either allow or not allow users to change 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 do not allow the users to change 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 Change at First Login or Reset

The Directory Server password policy lets you decide whether users must change their passwords at the first login or after the password is reset by an 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 the previous section on "User-Defined Passwords".)

By default, users do not need to change their passwords at first login or after a reset.

Password Expiration

You can configure your password policy so that users can use the same passwords indefinitely, or 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. However, 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.

Directory Server remembers the password expiration configuration even if you disable password expiration. This means that if you re-enable password expiration, 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 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.

The new global configuration attribute, `usePwdChangedTime`, enables you to limit the duration during which a user can log in after a password is changed, for example, after a password is reset by an administrator.

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. Directory Server displays the warning when the user binds to the server. By default, if password expiration is turned on, a warning is sent (via an 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 syntax guidelines for password strings. The password syntax-checking mechanism ensures that password strings conform to these guidelines. By default, password syntax checking is turned off. Password length is checked, only if password syntax checking is enabled.

Password Length

Directory Server enables 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 so that users can remember the password without writing it down.

By default the minimum password length is 6 characters. The minimum length of a password is checked only if password syntax checking is turned on.

Password Minimum Age

You can configure your password policy to prevent users from changing their passwords with a specified period. You can use this feature in conjunction with the password history feature to discourage users from reusing old passwords.

Setting the password minimum age to 2 days, for example, prevents users from repeatedly changing their 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 configure Directory Server to store a maximum of 24 used passwords in history.

With password history enabled, if a user attempts to reuse one of the passwords Directory Server has stored, the password is rejected. This feature prevents users from reusing one or two passwords that are easy to remember.

If you disable the password history feature, no previously used passwords are stored and users are able to reuse passwords. By default, Directory Server does maintain a password history.

Password Storage Scheme

The password storage scheme specifies the type of encryption used to store 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 method.
- 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 clear text passwords in the directory. The CRYPT algorithm provides compatibility with UNIX passwords. SSHA is the most secure, and is the default hash algorithm for Directory Server.

Configuring Password Policies

Directory Server 5.2 provides the following password policy options:

- A *global password policy*, stored under `cn=Password Policy,cn=config`, is applied by default. The parameters of this global policy can be changed. The global password policy was the only password policy that existed in Directory Server versions prior to 5.2. As a backup to the global password policy, a *hard-coded password policy* is provided. This policy is applied if the global password policy is absent or, following modifications, no longer valid. The attribute values of the hard-coded password policy are the same as those of the default global password policy.
- You can define a password policy and apply it to a particular user.
- You can define a password policy and apply it to a set of users using the CoS and Roles functionality.

This section describes these options in more detail, and explains the order of precedence that governs the application of password policies when multiple password policies exist for a given user entry. This section is divided into the following topics:

- Global Password Policy
- Defining Password Policies for Users or Sets of Users
- Multiple Password Policies and Their Order of Precedence

Global Password Policy

The global password policy is stored under `cn=Password Policy,cn=config`, and is applied by default. You can modify this policy to suit your security needs. The default global policy enforces the following:

- The SSHA storage scheme.
- Users can change their passwords.
- Users do not have to change their password at their first login or after the password is reset by an administrator.
- Password syntax checks (compliance with the minimum number of characters) are not performed.
- Passwords never expire.
- If you activate password expiration, the maximum age of a password is 100 days.
- No time needs to elapse between modifications to the password.
- If you activate password expiration, a password expiration warning is sent one day before the password is due to expire on the user’s first bind attempt.
- Used passwords are not recorded.
- Users are never locked out of their accounts.
- If you activate account lockout, users are locked out after a maximum number of three failed bind attempts, and the lockout lasts for one hour.
- Password failures are purged from the failure counter after 600 seconds.

The default policy (where passwords never expire, no syntax checks are performed, and the account lockout mechanism is not enabled) has low management overheads but is not the most secure. You should balance your security requirements against the management overheads generated by a demanding password policy.

NOTE In previous versions of Directory Server, global password policy attributes were stored directly under `cn=config`. They are now stored under `cn=Password Policy,cn=config`. If this entry does not exist, the hard-coded password policy provided with Directory Server is applied.

Defining Password Policies for Users or Sets of Users

You define a password policy for a specific user entry or for a set of users, with the `passwordPolicySubentry` attribute. The value of this attribute is the DN of an LDAPsubentry that contains the password policy attributes you wish to apply directly to the user's entry. This attribute can either be a real attribute or a virtual attribute generated by a CoS definition.

You can define password policies for a set of users by configuring the CoS definition to provide values for the `passwordPolicySubentry` attribute in user entries as a function of the Roles that those user entries have. While this is not the only way to define password policies for sets of users, it is the method used by Directory Server Console.

To make managing your password policy easier, you should co-locate the user or set of users to which the password policy applies, and the password policy itself. This will ensure that the password policy LDAP subentry is replicated along with the password policy.

For procedural information on defining individual password policies, see "Managing Individual Password Policies," in Chapter 7 of the *Directory Server Administration Guide*.

Multiple Password Policies and Their Order of Precedence

There are three main rules of precedence that govern the application of password policies when a user entry has more than one password policy assigned to it. The rules are as follows:

1. A password policy generated by a CoS definition will take precedence over a password policy assigned directly to the user entry. This is because the `cosAttribute` value defined in the CoS definition entry is obliged to contain an `operational` qualifier, which causes the CoS generated password policy to override any real attributes that may have been assigned directly to the user. For more information about the Roles and CoS mechanism, see Chapter 4, "The Directory Information Tree."

2. A password policy assigned directly to a user entry will take precedence over the global password policy.

3. The global password policy, stored under `cn=Password Policy,cn=config`, will take precedence over the hard-coded password policy values provided with Directory Server.

CAUTION When you are configuring password policies using CoS, it is important to establish an order of precedence in case a user entry is affected by more than one CoS generated password policy. You specify order of precedence by entering the appropriate value in the cosPriority attribute when you create your CoS template entry. You assign the highest priority with a value of 0. CoS templates that contain no cosPriority attribute are considered lowest priority, and when templates have the same (or no) cosPriority attribute value, a priority is chosen arbitrarily. For more information on Roles and CoS, see Chapter 4, "The Directory Information Tree."

Preventing Dictionary-Style Attacks

In a dictionary-style attack, an intruder attempts to crack a password by repeatedly guessing from a list of common words and variations, until gaining authorization. The server provides three tools to counter such attacks:

- Password syntax checking verifies that a password does not match values of the uid, cn, sn, givenName, ou, or mail attributes of the user entry. The server will not allow a user to set a password if it matches one of these values. However, syntax checking does not thwart actual dictionary attacks, in which the intruder tries every word in /usr/dict/words, for example.
- A minimum password length ensures that the user cannot set a short password. Passwords with more characters are exponentially harder to guess or attempt all values. In Directory Server, you must enable password syntax checking and minimum length simultaneously.
- The account lockout mechanism prevents binding after a certain number of failed authentication attempts. The lockout may either be temporary or permanent, depending on how strict you wish to make your password policy.

  Both will effectively prevent automated guessing of passwords. For example, if you allow five attempts and then lock the user account for five minutes, the intruder can make only one guess per minute, on average, and a poor typist is inconvenienced only momentarily. If the lock is permanent, the user's password must be reset manually by the Directory Manager.

  Account lockout counters are local to a directory server. This feature is not designed as a global lockout from your directory service, which means that even in a replicated environment, account lockout counters are not replicated.

Password Policies in a Replicated Environment

Configuration information for the global password policy is *not* replicated, because it is an entry under `cn=config`. If you modify the global password policy, you must therefore make the same modifications on each of the servers in your topology manually. If you require a global password policy that *is* replicated, you must define such a policy under a part of the directory tree that is replicated.

All password information that is stored in the user entry (current password, password history, password expiration dates, and so forth) is replicated. Account lockout counters are stored at the local server level and are *not* replicated.

You should consider the following impact of password policies in a replicated environment:

- A user with an impending password expiration will receive a warning from every replica to which they bind before changing their password.
- When a user changes their password, it may take time for this information to be updated on all replicas. If a user changes a password and then immediately rebinds to one of the consumer replicas with the new password, the bind may fail until the replica receives the updated password.
- Each replica keeps separate, non-replicated account lockout counters. As a result, the lockout policy will be enforced on any single replica, but may be circumvented when a user attempts to bind to several replicas. For example, if you have 10 servers in the replication topology, and lock out is activated after three attempts, an intruder could potentially try 30 guesses of the password.

  While replication does allow an intruder more guesses, the number is insignificant when compared to the billions of password values. It is much more important to force users to have strong passwords by turing on password checking and setting a password length of six characters or more. You should also give them guidelines on how to select and remember a password that is not a common dictionary word. Finally, you should ensure that all directory administrator users have very strong passwords.
- In an environment that uses multiple password policies, you must replicate the LDAP subentry containing the policy definition to be applied to the replicated entries. If you fail to do so, the LDAP subentry containing the policy definition will not exist and the default password policy will be applied.

- Entries that are created for replication (for example, the server identity Replication Manager entries) must have passwords that never expire. To make sure that these special users have passwords that do not expire, add the `passwordExpirationTime` attribute to the entry and give it a value of `20380119031407Z` (the maximum value in the valid range).

Designing Access Control

Access control enables you to specify that certain clients have access to particular information, while other clients do not. You implement access control using one or more access control lists (ACLs). ACLs consist of a series of access control instructions (ACIs) that either allow or deny permissions (such as read, write, search, proxy, add, delete, and compare) to specified entries and their attributes.

Using an 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 group, or for all users of the directory. You can also define access for a network location such as an IP address or a DNS name.

This section describes the Directory Server access control mechanism, and is divided into the following topics:

ACI Format

ACIs are stored in the directory, as attributes of entries. The `aci` attribute is an operational attribute; it is available for use on every entry in the directory, regardless of whether it is defined for the object class of the entry. It is used by Directory Server to evaluate what rights are granted or denied when it receives an LDAP request from a client. The `aci` attribute is returned in an `ldapsearch` operation if specifically requested. For more information on the format of ACIs see “ACI Syntax,” in Chapter 6 of the *Directory Server Administration Guide*.

Default ACIs

When you install Directory Server, or when you add a new suffix, a number of default ACIs are defined. These ACIs can be modified to suit your security requirements. For details on the default ACIs and how to modify them, refer to “Default ACIs” in Chapter 6 of the *Directory Server Administration Guide*.

Setting Permissions

If no ACIs are present in the directory, the default policy is *not* to grant users access rights (with the exception of the Directory Manager.) Therefore, you must set some ACIs to enable your users to access the directory. The following sections describe the access control mechanism provided by Directory Server. For information on setting ACIs, see “Creating ACIs From the Command Line” and “Creating ACIs Using the Console” in Chapter 6 of the *Directory Server Administration 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. This 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 Directory Server, 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 Directory Server, 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.

Allowing or Denying Access

You can explicitly allow or deny access to your directory tree. Be careful of explicitly denying access to Directory Server. Because of the precedence rule, if the directory finds rules explicitly forbidding access, it 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 access, you avoid the need to set explicit deny access.

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 take 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 to 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 enable you to set access to any directory entry that matches a set of defined criteria.

For example, you could allow read access for any entry that contains an `organizationalUnit` attribute with a value of Marketing.

Filtered access control rules let you use predefined 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 with ACIs, see “Targeting Entries or Attributes Using LDAP Filters” in Chapter 6 of the *Directory Server Administration Guide.*

Using Macro ACIs

Macro ACIs are placeholders that are used to represent a DN, or a portion of a DN, in an ACI. You can use a macro to represent a DN in the target portion of the ACI, in the bind rule portion, or both. When Directory Server receives an LDAP request, the macro ACIs 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. Directory Server then evaluates the ACI normally.

When the server starts, all ACIs are brought into memory, although cache limits do not apply to them. This can have a considerable impact on memory consumption. If you use repeating directory tree structures, you should therefore optimize the number of ACIs used in Directory Server by using macro ACIs where possible.

For more information on macro ACIs see “Advanced Access Control: Using Macro ACIs” in Chapter 6 of the *Directory Server Administration Guide*.

Requesting Effective Rights Information

The access control model provided by Directory Server is powerful in that access can be granted to users via many different mechanisms. However, this flexibility can make determining what your security policy comprises fairly complex. Because there are several parameters that can define the security context of a user (IP address and machine name, time of day, and authentication method, for example,) it is useful to be able to list the rights of a given user to directory entries and attributes.

Directory Server enables you to request the effective access rights that a user has to specified directory entries and attributes. The effective rights information obtained corresponds to:

- The ACIs effective at the time of your request
- The authentication method used
- The host machine name and address from which you make the request

When establishing why a user does or does not have access to certain data, you must reflect all of the user’s parameters when initiating your effective rights search operation.

This section examines the effective rights feature in more detail and is divided into the following topics:

- About the Effective Rights Feature
- Access Control on the Effective Rights Feature
- Results of an Effective Rights Request

About the Effective Rights Feature

The effective rights feature uses the `ldapsearch` utility supplied with the Directory Server Resource Kit (DSRK). The rights information you require is specified using a particular option and the information relative to these rights is returned with your `ldapsearch` results. For information on how to use the effective rights feature, see "Viewing Effective Rights" in Chapter 6 of the *Directory Server Administration Guide*.

Access Control on the Effective Rights Feature

To obtain effective rights information, users must have the access control rights to use the Effective Rights control *and* read access to the `aclRights` attribute. This double layer of access control for the effective rights feature provides basic security which can be more finely tuned if necessary. By analogy with proxy, if you have read access to the `aclRights` attribute in an entry, you can request information about anyone's rights to that entry and its attributes. This implies that the user who manages the resource can determine who has rights to that resource, even if that user does not actually manage those with the rights.

If a user requesting rights information does not have the rights to use the Effective Rights control, the operation will fail and an error message will be returned. However, if the user requesting rights information does have the rights to use the control but lacks the rights to read the `aclRights` attribute, the `aclRights` attribute will simply be absent from the returned entry. This behavior reflects Directory Server's general search operation behavior.

If a proxy control is attached to an Effective Rights control-based search operation, the effective rights operation will be authorized as the proxy user. This means that the *proxy user* will require the rights to use the Effective Rights control, and the entries returned will be those entries that the *proxy user* has the right to search and view.

Results of an Effective Rights Request

The information returned as a result of an effective rights request includes :

- Rights information, including entry level rights, attribute level rights and logging.
- Write, selfwrite add, and selfwrite delete permissions.
- Logging information, which enables you to debug access control problems.

For detailed information on each of these aspects, refer to “Understanding Effective Rights Results,” in Chapter 6 of the *Directory Server Administration Guide*.

Tips on Using ACIs

The following tips can help to lower the administrative burden of managing your directory security model and improve directory performance. Some of the following hints have already been described earlier in this chapter, but are included here to provide a complete list.

- Minimize the number of ACIs in your directory and use macro ACIs where possible.

  Although Directory Server can evaluate over 50,000 ACIs, it is difficult to manage a large number of ACI statements, and excessive ACIs may have a negative impact on memory consumption.

- 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 prevent excessive allow ACIs close to the leaf entries.

- Identify the smallest set of attributes on any given ACI.

  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 denies write access for those few attributes.

- Use LDAP search filters cautiously.

Because search filters do not directly name the object for which you are managing access, 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 to just the `commonName` attribute, 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 ACIs from Directory Server Console.

- Use standard attributes in user entries to determine access rights.

  As far as possible, use information that is already part of standard user entries to define access rights. If you need to create special attributes, consider creating them as part of a role or Class of Service (CoS) definition. For more information on roles and CoS, refer to "Grouping Directory Entries and Managing Attributes" on page 65.

- 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 the total list of ACIs, and also 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 to the server, it can be confusing for an administrator.

ACI Limitations

When creating an access control policy, be aware of the following restrictions:

- If your directory tree is distributed over several servers using the chaining feature, some restrictions apply to the keywords you can use in access control statements:
  - ACIs that depend on group entries (`groupdn` keyword) must be located on the same server as the group entry. If the group is dynamic, all members of the group must have an entry on the server too. If the group is static, the members' entries can be located on remote servers.
  - ACIs that depend on role definitions (`roledn` keyword) must be located on the same server as the role definition entry. Every entry that is intended to have the role must also be located on the same server.

  However, you can do value matching of values stored in the target entry with values stored in the entry of the bind user (for example, using the `userattr` keyword). Access is evaluated normally even if the bind user does not have an entry on the server that holds the ACI.

  For more information on how to chain access control evaluation, see "Access Control Through Chained Suffixes" in Chapter 3 of the *Directory Server Administration Guide*.

- Attributes generated by a CoS cannot be used in all ACI keywords. Specifically, you should not use attributes generated by CoS with the `userattr` keyword because the access control rule will not work.
- Access control rules are always evaluated on the local server. Therefore, it is not necessary to specify the hostname or port number of the server in LDAP URLs used in ACI keywords. If you do, the LDAP URL will not be taken into account. For more information on LDAP URLs, see LDAP URL Reference in the *Directory Server Administration Reference*.
- The cache settings used for ensuring that the server fits the physical memory available *do not* apply to ACI caches, which means that an excessive number of ACIs may saturate available memory.
- When granting proxy rights, you cannot grant a user the right to proxy as the Directory Manager, nor can you grant proxy rights to the Directory Manager.

Securing Connections With SSL

In addition to designing an authentication scheme for identifying users and an access control scheme for protecting information, you must protect the integrity of the information in transit over the network between servers and client applications.

To provide secure communications over the network you can use both the LDAP and DSML-over-HTTP protocols over the Secure Sockets Layer (SSL). When you have configured and activated SSL, clients connect to a dedicated secure port where all communications are encrypted once the SSL connection is established. Directory Server also supports the Start Transport Layer Security (Start TLS) control, which allows the client to initiate an encrypted connection over the standard LDAP port. For a comprehensive overview of SSL and TLS, refer to Chapter 9, “Using SSL and TLS with Sun Java System Servers,“ in the *Administration Server Administration Guide*.

Directory Server supports SSL-secured connections and non-SSL connections simultaneously.

SSL uses encryption for privacy and hashing of checksums for data integrity. When establishing an SSL connection, the client application and Directory Server select the strongest encryption algorithm (*cipher*) common to their configurations. Directory Server may use any of the following ciphers:

- DES - 56-bit block cipher
- 3DES (“triple-DES”) - 156-bit block cipher
- RC2 - 128-bit block cipher (or 40-bit export cipher)
- RC4 - 128-bit stream cipher (or 40-bit export cipher)

Ciphers are combined with one of the following hashing algorithms:

- MD5
- SHA-1

After a secure connection has been established, the SSL protocol requires the server to send its certificate to the client. Using public-key cryptography, the client can determine the authenticity of the certificate and verify that it was issued by a certificate authority that the client trusts. By verifying the certificate, the client can prevent a *man-in-the-middle* impersonation of the server by a third party.

You can also configure the server to request authentication from the client. Directory Server supports certificate-based and SASL-based client authentication. These mechanisms are described in "Selecting Appropriate Authentication Methods" on page 146. Client authentication to the server provides the highest level of security by ensuring that no third party may intercept or interfere with the communication between the client and the server.

To enhance the performance for connections using the SSL protocol with certificate-based authentication, Directory Server 5.2 supports the Sun Crypto Accelerator Board. This board accelerates SSL key-related calculations, and may be useful in deployments where client applications repeatedly bind over SSL, search, and then unbind. SSL accelerator boards may not improve Directory Server performance when key-related calculations are not the performance bottleneck. In addition, SSL accelerator boards are most effective if the clients that are establishing connections are doing so from different machines. If a system establishes multiple SSL-based connections, it is likely that the SSL caching session will limit the number of RSA operations, which will in turn limit the benefit that the accelerator board may provide. For information on how to install and configure the Sun Crypto Accelerator Board see Appendix A, "Using the Sun Crypto Accelerator Board" in the *Directory Server Administration Guide*.

For information about configuring and enabling SSL in Directory Server and its clients, refer to Chapter 11, "Managing Authentication and Encryption" in the *Directory Server Administration Guide*.

Encrypting Attributes

This section presents the attribute encryption functionality, and is divided into the following topics:

- What is Attribute Encryption?
- Attribute Encryption Implementation
- Attribute Encryption and Performance
- Attribute Encryption Usage Considerations

What is Attribute Encryption?

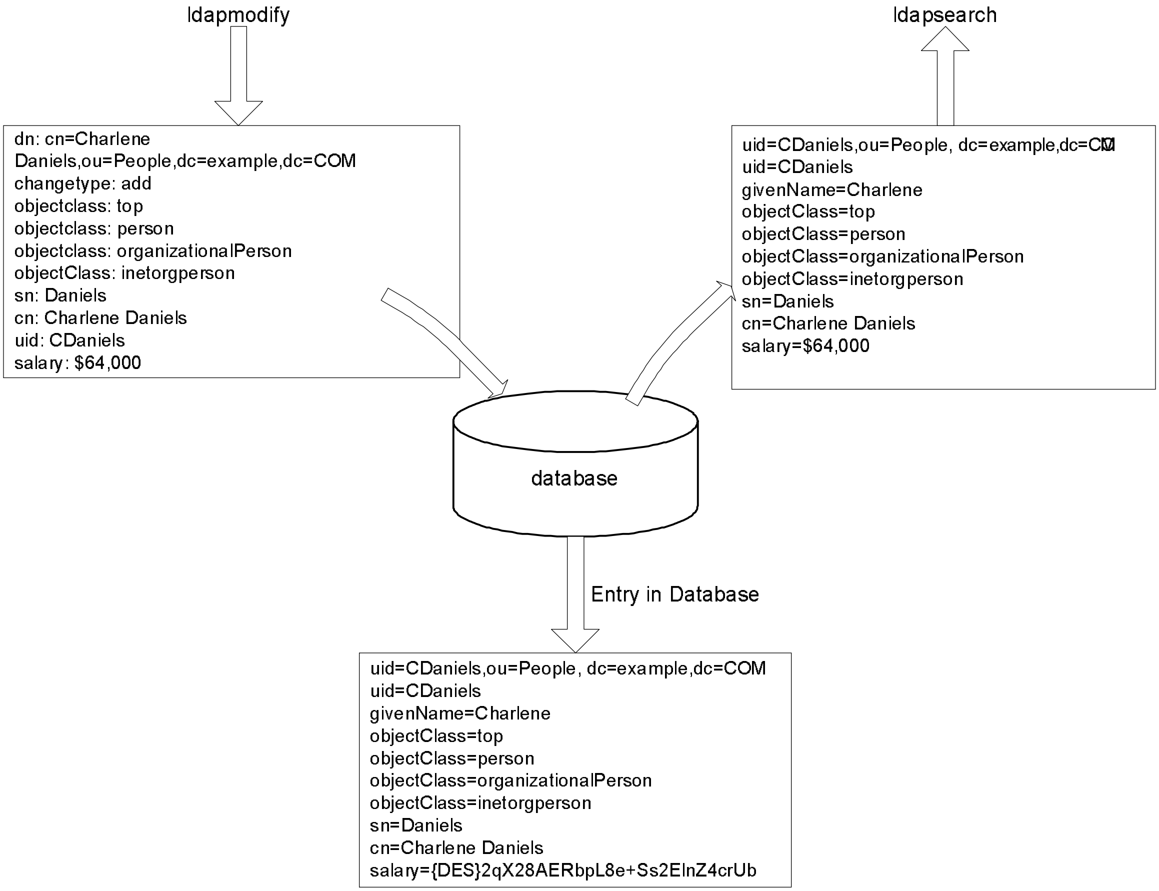
Directory Server provides a variety of features to protect data at access level (during reads and writes to the directory), including simple password authentication, certificate-based authentication, Secure Sockets Layer (SSL), and proxy authorization. However, there is often an additional need for the data stored in database files, backup files, and LDIF files to be protected. Consider a bank storing 4-digit PIN codes in the directory. If the database files were unprotected and were dumped, unauthorized users could have access to this sensitive information. The attribute encryption feature prevents users from accessing sensitive data while it is in storage.

Attribute encryption enables you to specify that certain attributes be stored in an encrypted form. It is configured at the database level, which means that once you decide to encrypt an attribute, that attribute will be encrypted for every entry in the database. Because attribute encryption occurs at an attribute level rather than an entry level, the only way to encrypt an entire entry is to encrypt all of its attributes.

In addition to protecting data while in storage, attribute encryption enables you to export data to another database in encrypted format. However, because the purpose of attribute encryption is to protect sensitive data only when it is in storage or being exported, the encryption is always reversible. Encrypted attributes are therefore decrypted when returned via search requests.

Figure 7-1 shows a user entry being added to the database, where attribute encryption has been configured to encrypt the `salary` attribute.

Figure 7-1 Attribute Encryption Logic



```
dn: cn=Charlene
Daniels,ou=People,dc=example,dc=COM
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectClass: inetorgperson
sn: Daniels
cn: Charlene Daniels
uid: CDaniels
salary: $64,000
```

```
uid=CDaniels,ou=People, dc=example,dc=COM
uid=CDaniels
givenName=Charlene
objectClass=top
objectClass=person
objectClass=organizationalPerson
objectClass=inetorgperson
sn=Daniels
cn=Charlene Daniels
salary=$64,000
```

```
uid=CDaniels,ou=People, dc=example,dc=COM
uid=CDaniels
givenName=Charlene
objectClass=top
objectClass=person
objectClass=organizationalPerson
objectClass=inetorgperson
sn=Daniels
cn=Charlene Daniels
salary={DES}2qX28AERbpL8e+Ss2ElnZ4crUb
```

Attribute Encryption Implementation

The attribute encryption feature supports a wide range of encryption algorithms, and ensures portability across different platforms. As an additional security measure, attribute encryption uses the private key of the server's SSL certificate to generate its own key, which is used to perform the encryption and decryption operations. This implies that, in order to be able to encrypt attributes, your server must be running over SSL. The SSL certificate and its private key are stored securely in the database in that they are protected by a password, and it is this key database password that is required to authenticate to the server. It is assumed that whoever has access to this key database password will be authorized to export decrypted data.

When importing data online with a view to encrypting it, you will already have provided the key database password to authenticate to the server, and will not be prompted a second time. If you are importing data offline, Directory Server will prompt you for the password before it allows you to encrypt the data you are importing. When decrypting data (a more security sensitive operation), Directory Server automatically prompts you for the key database password, regardless of whether the export operation is online or offline. This provides an additional security layer.

NOTE As long as the certificate or private key does not change, the server will continue to generate the same key, which will make it possible to transport (export then import) data from one server instance to another (provided both server instances have used the same certificate.)

Attribute Encryption and Performance

While attribute encryption offers increased data security, it does incur certain performance costs. Bearing this in mind, you should think carefully about which attributes require encryption and encrypt only those attributes you consider to be particularly sensitive.

Because sensitive data can be accessed directly through index files, it is necessary to encrypt the index keys corresponding to the encrypted attributes, to ensure that the attributes are fully protected. Given that indexing already has an impact on Directory Server performance (without the added cost of encrypting index keys), it is advisable to configure attribute encryption *before* data is imported or added to the database for the first time. This procedure will ensure that encrypted attributes are indexed as such from the outset.

Attribute Encryption Usage Considerations

Consider the following when implementing the attribute encryption feature:

- As a general best practice when modifying attribute encryption configuration, you should export your data, make the configuration changes, and then import the newly configured data.

  This will ensure that all configuration changes are taken into account in their entirety, without any loss in functionality. Failing to do so could result in some functionality loss and thus compromise the security of your data.

- Modifying attribute encryption configuration on an existing database can have a significant performance impact.

  Imagine for example that you have a database instance with existing data. The database contains previously stored entries with a sensitive attribute called `mySensitiveAttribute`. The value of this attribute is stored in clear text, in the database and in the index files. If you decide to encrypt the `mySensitiveAttribute` attribute, all the data in the database instance must be exported and re-imported into the database to ensure that the server updates the database and index files taking the attribute encryption configuration into account. This will have a significant performance impact that could have been avoided had the attribute been encrypted from the beginning.

- When exporting data in decrypted format the export will be refused if an incorrect password is given.

  As a security measure, the server prompts users for passwords if they want to export data in decrypted format, and refuses the decrypted export operation should users provide an incorrect password. Passwords can be entered directly or by providing the path to a file containing the password (which has the same syntax as the SSL password file.)

- Algorithm changes are supported, but can result in lost indexing functionality, if they are not made correctly.

  To change the algorithm used to encrypt data, export the data, modify the attribute encryption configuration, and then import the data, to avoid any functionality loss related to indexing. If you do not follow this procedure, the indexes created on the basis of the initial encryption algorithm will no longer function.

  Because the encrypted attributes are prefaced with a cipher tag that indicates the encryption algorithm used, the internal server operations take care of importing the data. Directory Server therefore enables you to export data in encrypted form before making the algorithm change.

- Changing the server's SSL certificate will result in you no longer being able to decrypt encrypted data.

  Because the server's SSL certificate is used by the attribute encryption feature to generate its own key, which is then used to perform the encryption and decryption operations, this certificate is required to decrypt encrypted data. Changing the certificate without decrypting the data beforehand will result in you no longer being able to decrypt the data. To avoid this, you should export your data in decrypted format, changing the certificate and then re-import the data.

- To transport data in encrypted format, that is, export and import it from one server instance to another, both server instances must use the same certificate.

For procedural information on the attribute encryption feature, see "Encrypting Attribute Values" in Chapter 2 of the *Directory Server Administration Guide*.

Grouping Entries Securely

This section deals with the security issues related to grouping entries and contains the following topics:

- Using Roles Securely
- Using CoS Securely

Using Roles Securely

Not every role is suitable for use within a security context. When creating a new role, consider how easily the role can be assigned to and removed from an entry. Sometimes it is appropriate for users to be able to add themselves to or remove themselves from a role. For example, if you had an interest group role called Mountain Biking, you would want interested users to add themselves or remove themselves easily.

However, in some security contexts it is inappropriate to have such open roles. For example, consider account inactivation roles. By default, account inactivation roles contain ACIs defined for their suffix (for more information about account inactivation, see "Inactivating Users and Roles" in Chapter 7 of the *Directory Server Administration Guide*.) When creating a role, the server administrator decides whether or not a user can assign themselves to or remove themselves from the role.

For example, user A possesses the managed role, MR. The MR role has been locked using account inactivation through the command line. This means that user A cannot bind to the server because the `nsAccountLock` attribute is computed as “true” for that user. However, suppose the user was already bound and noticed that he is now locked through the MR role. If there are no ACIs preventing him, the user can remove the `nsRoleDN` attribute from his entry and unlock himself.

To prevent users from removing the `nsRoleDN` attribute, you would need to apply ACIs. With filtered roles you would have to protect the part of the filter that would prevent the user from being able to relinquish the filtered role by modifying an attribute. Users should not be allowed to add, delete, or modify the attribute used by the filtered role, and in the same way if the value of the filter attribute is computed then all the attributes that can modify the value of the filter attribute should be protected too. As nested roles can be comprised of filtered and managed roles, the above points should be considered for each of the roles that comprise the nested role.

Using CoS Securely

Access control for reading applies to both the real and virtual attributes of an entry. A virtual attribute generated by the Class of Service mechanism is read just as a normal attribute and should be given read protection in the same way.

However, in order to make the CoS value secure, you must protect all of the sources of information it uses: the definition entries, the template entries, and the target entries. The same is true for update operations: write access to each source of information should be controlled to protect the value that is generated from them.

The following sections describe the general principals for read and write protection of data in each of the CoS entries. The detailed procedure for defining individual access control instructions (ACIs) is described in Chapter 6, “Managing Access Control” in the *Directory Server Administration Guide.*

Protecting the CoS Definition Entry

Although the CoS definition entry does not contain the value of the generated attribute, it provides the information to find that value. Reading the CoS definition entry would reveal how to find the template entry containing the value, and writing to this entry would modify how the virtual attribute is generated.

You should therefore define both read and write access control on the CoS definition entries.

Protecting the CoS Template Entries

The CoS template entry contains the value of the generated CoS attribute. Therefore, as a minimum, the CoS attribute in the template should be protected for both reading and updating.

In the case of pointer CoS, there is a single template entry that should not be allowed to be renamed. In most cases, it is simplest to protect the entire template entry.

With classic CoS, all template entries have a common parent given in the definition entry. If only templates are stored in this parent entry, access control to the parent entry will protect the templates. However, if other entries beneath the parent require access, the template entries must be protected individually.

In the case of indirect CoS, the template may be any entry in the directory, including user entries that might still need to be accessed. Depending on your needs, you can either control access to the CoS attribute throughout the directory, or choose to ensure that the CoS attribute is secure in each entry used as a template.

Protecting the Target Entries of a CoS

All entries in the scope of a CoS definition, for which the virtual CoS attribute will be generated, also contribute to computing its value.

When the CoS attribute already exists in a target entry, by default, the CoS mechanism will not override this value. If you do not want this behavior, you should either define your CoS to override the target entry or protect the CoS attribute in all potential target entries. For information on these procedures see Chapter 5, “Managing Identity and Roles” in the *Directory Server Administration Guide*.

Both indirect and classic CoS also rely on a specifier attribute in the target entry. This attribute gives the DN or RDN of the template entry to use. You should protect this attribute either globally throughout the scope of the CoS or individually on each target entry where needed.

Protecting Other Dependencies

It is possible to define virtual CoS attributes in terms of other generated CoS attributes and roles. You will need to understand and protect these dependencies in order to guarantee the protection of your virtual CoS attribute.

For example, the CoS specifier attribute in a target entry could be `nsRole`, and therefore the role definition would also need to be protected. For more information, see “Grouping Entries Securely” on page 176.

In general, any attribute or entry that is involved in the computation of the virtual attribute value should have both read and write access control. For this reason, complex dependencies should be well planned or simplified to reduce subsequent complexity of access control implementation. Keeping dependencies on other virtual attributes to a minimum also improves directory performance and reduces maintenance.

Securing Configuration Information

For the majority of deployments no additional access controls are required either for the root DSE entry (the entry returned for a base object search with a zero-length DN) or for the subtrees below `cn=config`, `cn=monitor` or `cn=schema`. The root DSE entry and these subtrees contain attributes that are automatically generated by Directory Server and used by LDAP clients to determine the capabilities and configuration of the directory server.

However, one of the root DSE entry attributes called `namingContexts` contains a list of the base DNs for each of the Directory Server databases. In addition to this list, these DNs are also stored in the mapping tree entries below `cn=config` and `cn=monitor`. Should you wish to hide the existence of one or more subtrees and protect your configuration information for security reasons, it will be necessary to place:

- An ACI attribute in the entry at the base of the subtree you wish to hide.
- An ACI in the root DSE entry on the `namingContexts` attribute.
- An ACI on the `cn=config` and `cn=monitor` subtrees.

Other Security Resources

For more information about designing a secure directory, see the following:

- Sun Developer Security Resources
  http://developers.sun.com/techtopics/security/index.html
- *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 Server Monitoring

An effective monitoring and event management strategy is crucial to any successful Directory Server deployment. Such a strategy defines which events should be monitored, which tools to use, and what action to take should an event occur. Having a plan for common-place events helps prevent possible outages and reduced levels of service, improving the availability and quality of service.

A monitoring and event management strategy should include specific components of the architecture such as the replication configuration, but should also include system and network monitoring. This chapter examines what an effective monitoring strategy should include, and presents the monitoring features within Directory Server.

NOTE This chapter does *not* focus on system and network monitoring, as this is an area not specific to Directory Server.

This chapter is divided into the following sections:

Defining a Monitoring and Event Management Strategy

This section provides an outline of the stages involved in defining a monitoring and event management strategy. The process can be broken down into the following steps:

1. Select the appropriate monitoring tools, whether they be operating system tools, Directory Server monitoring tools, or third party monitoring tools.
2. Identify the key areas to be monitored in the directory architecture (these are frequently the same as the sizing and tuning attributes).
3. Define what triggers an event or alarm condition when monitoring the key performance measure. This implies defining an acceptable level of performance or operation for each performance measure.
4. Determine what action should be taken when an alarm condition occurs.

Directory Server Monitoring Tools

This section provides a summary of the monitoring tools available in Directory Server, and other tools that can be used to monitor Directory Server activity. All of the key performance measures, described in the next section, can be monitored using one, or a combination of, these tools.

- Command-Line Tools

  Command-line monitoring tools include operating system-specific tools to monitor performance such as disk usage, LDAP tools such as `ldapsearch` to collect server statistics stored in the directory, third party tools, or custom shell or Perl scripts.

- Directory Server logs

  The access, audit, and error logs provided with Directory Server are a rich source of monitoring information. These logs can be monitored manually or parsed using custom scripts to extract monitoring information relevant to your deployment. The Directory Server Resource Kit provides a log analyzer tool, `logconv.pl`, that enables you to analyze Directory Server access logs. The log analyzer tool extracts usage statistics and counts the occurrences of significant events. For more information this tool, refer to Chapter 24, ”The Log Analyzer Tool,” in the *Directory Server Resource Kit Tools Reference*. For information on viewing and configuring log files refer to Chapter 13, “Monitoring Directory Server Using Log Files” chapter in the *Directory Server Administration Guide*.

- Directory Server Console

  Directory Server Console enables you to monitor directory operations in real time, via a graphical user interface. The Console provides general server information, including a resource summary, current resource usage, connection status, and global database cache information. It also provides general database information such as the database type, status and entry cache statistics, cache information, and information relative to each index file within the database. In addition, the Console provides information relative to the connections and operations performed on each chained suffix.

- Replication Monitoring Tools

  The replication monitoring tools provided with Directory Server enable you to:

  - monitor the state of synchronization between a master replica and one or more consumer replicas
  - compare the same entry on two or more different replicas, enabling you to assess replication status
  - depict your complete replication topology, which is particularly beneficial when dealing with complex directory deployments

- Simple Network Management Protocol (SNMP)

  Directory Server supports monitoring with the Simple Network Management Protocol (SNMP). SNMP is the standard mechanism for global network control and monitoring, and enables network administrators to centralize network monitoring activity.

  For a detailed description of SNMP and Directory Server’s SNMP managed object support see “SNMP Monitoring” on page 191. For information on how to set up and configure SNMP refer to Chapter 14, “Monitoring Directory Server Using SNMP” in the *Directory Server Administration Guide*.

Directory Server Monitoring

The most important step in defining a monitoring and event management strategy is determining the key areas to be monitored on one or more components in your directory architecture. What you monitor, and to what extent, will depend largely on the specifics of your deployment.

This section describes the performance measures that should be monitored, and includes the following:

- Monitoring Directory Server Activity
- Monitoring Database Activity
- Monitoring Disk Status
- Monitoring Replication Activity
- Monitoring Indexing Efficiency
- Monitoring Security

NOTE

- When running `ldapsearch` commands on the monitoring information in the `cn=monitor` branch of the directory, users must be authenticated and have the appropriate permissions to access the information. Having such permissions is therefore a prerequisite in defining your monitoring strategy
- It is essential to monitor the operating system environment on which Directory Server is running, to ensure that the system is running efficiently and not compromising the server. However, this area is *not* covered in this chapter as it is not specific to Directory Server. Refer to your operating system documentation for further information.

Monitoring Directory Server Activity

Directory Server provides a number of ways in which you can monitor server status. These include, but are not limited to, the following:

- The Servers and Applications tab of Sun Java System Server Console displays general information regarding your server including the installation date, the version, the server status (whether or not it is started) and the port numbers.

- Directory Server Console provides access to additional monitoring information. The Status tab on this console displays the following information:
  - The startup and current time on the server.
  - A Resource Summary that details connections, initiated and completed operations, and entries and bytes sent to clients.
  - Current Resource Usage information, including active threads, open and available connections, number of threads waiting to read from the client, and number of databases in use.
  - Information on all Open Connections, including when they were opened, how many connections were started and completed, the distinguished name used by the client to bind to the server, the state of the connection (Blocked or Not blocked), and the type of connection (LDAP, or DSML.)

  For more information regarding the performance counters available through Directory Server Console, refer to “Monitoring Your Server Using the Console” in the *Directory Server Administration Guide*.
- Running an `ldapsearch` command on the `cn=monitor` entry provides access to the same information presented in the Status tab of Directory Server Console. Note that certain monitoring information is accessible only if the user issuing the `ldapsearch` command is bound as Directory Manager. You can remove this access constraint by reconfiguring the access control associated with this information. For details regarding the performance counters stored under `cn=monitor`, refer to “Monitoring Your Server From the Command Line” in the *Directory Server Administration Guide*.
- The `ps` command displays processes that are currently running. This enables you to determine whether the Directory Server `slapd` daemon is running. Refer to the `ps(1)` man page for more information.
- The `ldapsearch` command-line utility enables you to test whether Directory Server is responding to requests. To avoid launching time-consuming, unindexed searches, it is wise to use base level searches. Where you have more than one database, it is also wise to create an LDAP query for each database suffix to test whether or not the database is online and responding.
- Directory Server access logs enable you to monitor server operations and to establish whether the server is running. For more information on the access logs and connection codes refer to “Access Log Content,” and “Common Connection Codes,” in Chapter 3 of the *Directory Server Administration Reference*.

- The Directory Server error log records the server's start and stop status, and enables you to establish that the server is running. For more information about viewing and configuring log files refer to Chapter 13, "Monitoring Directory Server Using Log Files" in the *Directory Server Administration Guide.*

Monitoring Database Activity

Monitoring database activity helps to ensure that your database is online and accessible when it is required. Database monitoring information can be accessed by running an `ldapsearch` command on a specific area of the `cn=config` branch. The kind of monitoring information provided and the corresponding area of the `cn=config` branch are presented in Table 8-1.

Table 8-1 Source of Database Monitoring Information in cn=config

Information Area	**Corresponding Branch of cn=config**
General Database Information	`cn=database,cn=monitor,cn=ldbm database, cn=plugins,cn=config`
Database Cache Information	`cn=monitor,cn=ldbm database,cn=plugins,cn=config`
Specific Database Instance Information	`cn=monitor,cn=`*suffixName*`,cn=ldbm database, cn=plugins,cn=config`
Chained Suffix Information	`cn=monitor,cn=`*suffixName*`,cn=chaining database, cn=plugins,cn=config`

The areas of database monitoring information are presented in more detail in the following section.

- The `cn=database,cn=monitor,cn=ldbm database,dn=plugins,cn=config` branch provides access to cache, transaction, locks and log information. For a complete list of Directory Server configuration attributes, refer to Chapter 2, "Server Configuration Reference," in the *Directory Server Administration Reference*.

  The type of general database information you monitor will depend on the specific requirements of your directory deployment. For example, if your Directory Server frequently handles several simultaneous transactions, you may want to monitor the maximum number of transactions being handled at a particular time. If this number (defined by the `nsslapd-db-max-txns` attribute) approaches the maximum number of transactions allowed (defined by the `nsslapd-db-configured-txns` attribute), you may want to increase the maximum number of transactions allowed, to prevent operations from failing.

- To monitor database cache performance and database indexing performance, use the Status tab of Directory Server Console or run `ldapsearch` commands on the the following branches:

  `cn=monitor,cn=ldbm  database,cn=plugins,cn=config` and `cn=monitor,cn=`*suffixName*`,cn=ldbm database,cn=plugins,cn=config`

  For a complete list of the relevant configuration attributes refer to Chapter 2, “Server Configuration Reference,” in the *Directory Server Administration Reference*.

- The `cn=monitor,cn=`*suffixName*`,cn=chaining database,cn=plugins,cn=config` branch provides access to information about connections and the LDAP and bind / unbind operations being performed. This information is also accessible via the Status tab of Directory Server Console.

Monitoring Disk Status

Effectively monitoring disk space enables you to prevent the problems associated with inadequate disk resources. The `cn=disk,cn=monitor` entry provides access to the following monitoring information:

- The path to the database instance. Where several database instances reside on the same disk or an instance refers to several directories on the same disk, the short path name is displayed.
- The amount of disk space available to the server in MB.
- The status of the disk (normal, low or full). This status is based on the available space and on the thresholds configured to trigger a disk “low” and disk “full” warning. It is particularly important to monitor the disk full threshold, since the directory will no longer accept updates once this limit is reached.

For more information on the `cn=disk,cn=monitor` attributes as well as the configurable disk low or full thresholds, refer to Chapter 2, “Server Configuration Reference,” in the *Directory Server Administration Reference*.

Monitoring Replication Activity

Monitoring replication status is an essential element of your global monitoring strategy. The earlier you become aware of potential replication problems, the quicker you can resolve those problems and reestablish correct replication operation.

Directory Server 5.2 provides three replication monitoring tools which enable you to monitor various aspects of replication functionality. The replication monitoring tools function as LDAP clients and can be used over a standard or secure connection (LDAPS.) The following replication monitoring tools are provided:

- insync
- entrycmp
- repldisc

insync

The `insync` tool indicates the state of synchronization (or replication delay) between a master replica and one or more consumer replicas. This replication delay is an indication of how accurate the data is on a consumer, compared to the data on the master.

entrycmp

The `entrycmp` tool allows you to compare the same entry on two or more different servers. An entry is retrieved from the master replica and the entry's `nsuniqueid` is used to retrieve the same entry from a given consumer. Entry attributes and values are compared and, if these are identical, the entries are considered to be the same.

NOTE The machine on which you are running the `insync` and `entrycmp` tools must be able to reach all the specified hosts. If the hosts are unreachable due to a firewall, VPN, or other network setup reasons, you will encounter difficulties using these tools. For the same reason, you should ensure that all the servers are up and running before attempting to use the replication monitoring tools.

repldisc

The `repldisc` tool allows you to discover a replication topology. Topology discovery starts with one server and constructs a graph of all known servers within the topology. The `repldisc` tool then prints an adjacency matrix describing the topology. This replication topology discovery tool is useful for large, complex deployments where it might be difficult to recall the global topology you have deployed.

NOTES

- When using the replication monitoring tools, you must use either all symbolic names or all IP addresses when identifying hosts. Using a combination of the two can be problematic.
- When running the replication monitoring tools over SSL, the server on which you are running the tools must have a copy of all the certificates used by the other servers in the topology.
- These tools are based on LDAP clients, and as such, will need to authenticate to the server and use a bind DN that has read access to `cn=config`. For more information about the configuration details of these tools and using the tools with SSL enabled refer to Monitoring Replication Status in the *Directory Server Administration Guide*.

For more information about the replication monitoring tools, refer to “entrycmp,” “insync,” and “repldisc,” in Chapte r1 of the *Directory Server Administration Reference*.

Monitoring Indexing Efficiency

Indexing has a positive impact on read performance and a negative impact on write performance. It is therefore important to monitor indexing efficiency to maintain an appropriate balance between read and write performance. An effective indexing strategy eliminates unnecessary indexes and maintains only those indexes required for client applications.

Indexing efficiency can be monitored in the following ways:

- By consulting the access logs and monitoring the time unindexed searches take to complete, you can identify the unindexed searches that have taken a disproportionate amount of time. (Unindexed searches are identified in the log files by `notes=U` and long searches have a high value for `etime`.)

  The access log also provides additional information on searches and their filters, enabling you to decide whether it might be worth creating an index to improve performance. The Directory Server Resource Kit provides a log analyzer tool, `logconv.pl`, that enables you to analyze Directory Server access logs. For more information this tool, refer to Chapter 24, ”The Log Analyzer Tool,” in the *Directory Server Resource Kit Tools Reference*.

- The Status tab of Directory Server Console allows you to monitor the most frequently used indexes per suffix or chained suffix. It indicates how many attempts have been made to use the indexes and how many attempts have been successful. The same monitoring information can be accessed by running an `ldapsearch` command on the `cn=monitor,cn=`*suffixName*`,cn=ldbm database,cn=plugins,cn=config` branch.

  A list of configured indexes is available in the Configuration tab of Directory Server Console (under the Data > *suffixName* node). Comparing the frequently used indexes, described above, with the list of configured indexes enables you you to identify the indexes that are using resources unnecessarily, and to decide whether they can be removed. If entries contain indexed attributes and the indexes are not used, removing these indexes will improve add performance.

For more information on access log content and connection codes refer to “Access Log Content,” and “Common Connection Codes,” in Chapte r3 of the *Directory Server Administration Reference*. For a complete list of Directory Server configuration attributes, refer to Chapter 2, “Server Configuration Reference,” in the *Directory Server Administration Reference*.

Monitoring Security

Monitoring the security of your deployment is vital in maintaining a secure, accessible directory. Suggestions on how to monitor Directory Server with a view to maintaining an acceptable level of security follow:

- Monitoring the number of failed bind attempts alerts you to attempts to break into your directory. If the SNMP agent is running, failed bind attempts can be monitored by running an `ldapsearch` command on the SNMP managed object counter `dsBindSecurityErrors` located under `cn=snmp,cn=config`.

- Monitoring the number of open connections without any activity alerts you to potential denial of service attacks. The number of current connections and the number of completed operations can be accessed via the Status tab of Directory Server Console or by searching the attributes located under `cn=monitor`.
- The Effective Rights feature enables clients to query the access control rights they have to directory entries and attributes. Being able to request the access rights of a user simplifies user administration, access control policy verification, and configuration decision making.

  The Effective Rights feature would most likely be used periodically rather than on a day-to-day operations basis. For more detailed information regarding the Effective Rights feature see "Requesting Effective Rights Information" on page 165.

SNMP Monitoring

SNMP is the standard mechanism for global network control and monitoring. It allows network administrators to centralize network monitoring activities, and can be used to monitor a wide range of devices in real time. This section describes how SNMP can be used to monitor Directory Server operation, and contains the following topics:

- About SNMP
- SNMP Monitoring in Directory Server

About SNMP

SNMP is a protocol used to exchange data about network activity. With SNMP, data travels between a managed device and a network management station (NMS) where users manage the network remotely. A managed device is anything that runs SNMP, such as hosts, routers, and Directory Server. An NMS is usually a powerful workstation running one or more network management applications. A network management application usually displays graphical information about managed devices (which device is up or down, which and how many error messages were received, and so on).

Information is transferred between the NMS and the managed device through the use of two types of agents: the subagent and the master agent. The subagent gathers information about the managed device and passes the information to the master agent. Directory Server has a subagent. The master agent exchanges information between the various subagents and the NMS. The master agent runs on the same host machine as the subagents it talks to.

Multiple subagents can be installed on a host machine. For example, if Directory Server, Application Server, and Messaging Server are all installed on the same host, the subagents for each of these servers communicates with the same master agent. The master agent is installed with Administration Server.

Values for SNMP attributes that can be queried are kept on the managed device and reported to the NMS as necessary. Each attribute or variable is known as a managed object, which is anything the agent can access and send to the NMS. All managed objects are defined in a management information base (MIB), which is a database with a tree-like hierarchy. The top level of the hierarchy contains the most general information about the network. Each branch below is more specific and deals with a separate network area.

SNMP exchanges network information in the form of *protocol data units* (PDUs). PDUs contain information about variables stored on the managed device. These variables, also known as managed objects, have values and titles that are reported to the NMS as necessary. Communication between an NMS and a managed device takes place in one of two ways:

- NMS-Initiated Communication
- Managed Device-Initiated Communication

Directory Server supports NMS-initiated communication, described in the following section.

NMS-Initiated Communication

This is the most common type of communication between an NMS and a managed device. In this type of communication, the NMS either requests information from the managed device or changes the value of a variable stored on the managed device.

The following steps make up an NMS-initiated SNMP session:

1. The NMS determines which managed devices and objects must be monitored.

2. The NMS sends a protocol data unit to the managed device's subagent through the master agent. This protocol data unit either requests information from the managed device or tells the subagent to change the values for variables stored on the managed device.

3. The subagent for the managed device receives the protocol data unit from the master agent.

4. If the protocol data unit from the NMS is a request for information about variables, the subagent gives information to the master agent and the master agent sends it back to the NMS in the form of another protocol data unit. The NMS then displays the information textually or graphically.

   If the protocol data unit from the NMS requests that the subagent set variable values, the subagent sets these values.

SNMP Monitoring in Directory Server

Directory Server supports SNMP monitoring in two ways:

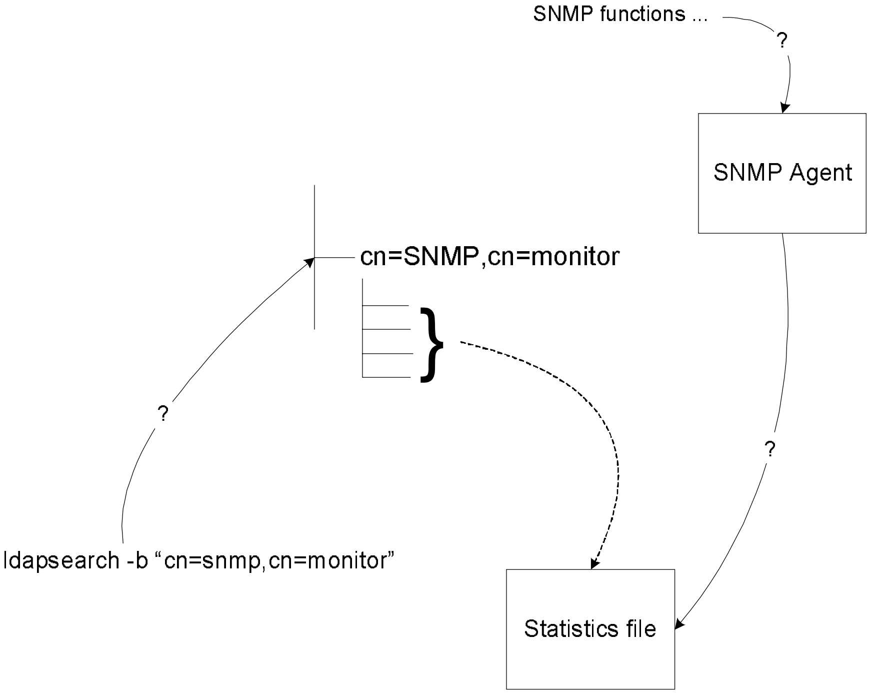
- Monitoring via an SNMP agent. SNMP attributes are mapped to a statistics file which is read each time the SNMP agent is queried. This statistics file is not present if Directory Server is not running.

- Monitoring using the `ldapsearch` command-line utility. SNMP attributes are stored under the `cn=snmp,cn=monitor` entry. The following `ldapsearch` command provides a list of all SNMP attributes in Directory Server:

  ```
  ldapsearch -h host -p port -s base -b "cn=snmp,cn=monitor"
   "objectclass=*"
  ```

Figure 8-1 shows the two ways in which SNMP monitoring information can be retrieved from Directory Server.

Figure 8-1 SNMP Monitoring in Directory Server



For information on where the MIBs are defined, and how to use SNMP refer to Chapter 14, "Monitoring Directory Server Using SNMP" in the *Directory Server Administration Guide*.

The SNMP managed objects supported by Directory Server are based on an early draft of the Directory Server Monitoring MIB RFC 2605. The SNMP operations managed objects returned by the SNMP agent are the same as the SNMP monitoring attributes returned by an `ldapsearch` command. These attributes are described in "Monitoring Attributes," in Chapter 2 of the *Directory Server Administration Reference*. Names of attributes returned by the SNMP agent are prefixed with `ds`.

In addition to the operations managed objects, Directory Server supports managed objects related to the interactions between the monitored server and its peer servers, and entity related managed objects, containing information about the current server installation. These objects are described in the “Interactions Table of Supported SNMP Managed Objects,” and the “Entity Table of SNMP Supported Managed Objects,“ in the *Directory Server Administration Reference*.

Chapter 9

Reference Architectures and Topologies

There are several factors to take into consideration when planning your directory deployment. Some of the most important considerations include the physical location of your data, how and where this data is replicated, what you can do to minimize failures, and how to react when failures do occur. The architectural strategies outlined in this chapter provide you with some guidelines.

This chapter is divided into the following sections:

- Addressing Failure and Recovery
- Planning a Backup Strategy
- Sample Replication Topologies

Addressing Failure and Recovery

It is essential to have a strategy in place for providing minimum disruption of service in the case of failure. For our purposes, failure is defined as anything that prevents Directory Server from providing the minimum level of service you require. This section describes the various reasons for which failure can occur, which will assist you in identifying and managing failures in your deployment.

Failure can be divided into two main areas:

- System unavailable
- System unreliable

The system may be *unavailable* due to any of the following:

- Network problem - the network may be down, slow or intermittent.
- Process (`slapd`) problem - the process may be down, busy, restarting, or unwilling to perform.
- Hardware problem - the hardware may be off, may have failed, or may be rebooting.

The system may be *unreliable* due to any of the following:

- Replication failure or latency, causing data to be out of date or unsynchronized.
- System too busy - an excess of read or write operations may result in unreliable data.

To maintain the ability to add and modify data in the directory, write operations should be routed to an alternative server in the event of a writable server becoming unavailable. Various methods can be used to reroute write operations, including the Sun Java System Directory Proxy Server.

To maintain the ability to read data in the directory, a suitable load balancing strategy must be put in place. Both software and hardware load balancing solutions exist to distribute read load across multiple consumer replicas. Each of these solutions also has the capability (to varying degrees of completeness and accuracy) to determine the state of each replica and to manage its participation in the load balancing topology.

Replicating directory contents increases the availability of Directory Server. A reliable replication topology will ensure that the most recent data is available to clients across data centers, even in the case of failure.

In the following sections, failure strategies for read and write operations are discussed as they relate to each replication topology.

Planning a Backup Strategy

In any failure situation involving data corruption or data loss, it is imperative that you have a recent backup of your data. If you do not have a recent backup, you will be required to re-initialize a failed master from another master. For a comprehensive set of procedures on how to back up your data, refer to Backing Up Data in the *Directory Server Administration Guide*.

This section provides a brief overview of what you should consider when planning a backup and recovery strategy.

Choosing a Backup Method

Directory Server provides two methods of backing up data: binary backup (`db2bak`) and backup to an LDIF file (`db2ldif`). These commands are both subcommands of the `directoryserver` command. See “directoryserver” in Chapter 1 of the *Directory Server Administration Reference* for more information. Both of these methods have advantages and limitations, and knowing how to use each method will assist you in planning an effective backup strategy.

Binary Backup (db2bak)

Binary backup is performed at the file system level. The output of a binary backup is a set of binary files containing all entries, indexes, the change log and the transaction log.

NOTE The `dse.ldif` configuration file is not backed up in a binary backup. You should back this file up manually to enable you to restore a previous configuration.

Performing a binary backup has the following advantages:

- All suffixes can be backed up at once.
- Binary backup is significantly faster than a backup to LDIF.

Binary backup has the following limitations:

- Restoration from a binary backup can be performed only on a server with an *identical* configuration. This implies that:
    - Both machines must use the same hardware and the same operating system, including any service packs or patches.
    - Both machines must have the same version of Directory Server installed, including binary format (32 bits or 64 bits), service pack and patch level.
    - Both servers must have the same directory tree divided into the same suffixes. The database files for all suffixes must be copied together, individual suffixes cannot be copied.
    - Each suffix must have the same indexes configured on both servers, including VLV (virtual list view) indexes. The databases for the suffixes must have the same name.
    - The Directory Server to be copied must not hold the o=NetscapeRoot suffix, which means it cannot be a configuration directory for Administration Server.
    - Each server must have the same suffixes configured as replicas, and replicas must have the same role (master, hub, or consumer) on both servers. If fractional replication is configured, it must be configured identically on all master servers.
    - Attribute encryption must not be used on either server.

  For more information on restoring data with the binary restore feature, refer to Initializing a Replica Using Binary Copy in the *Directory Server Administration Guide*.

At a minimum, you should perform a regular binary backup on each set of coherent machines (machines that have an identical configuration, as defined previously).

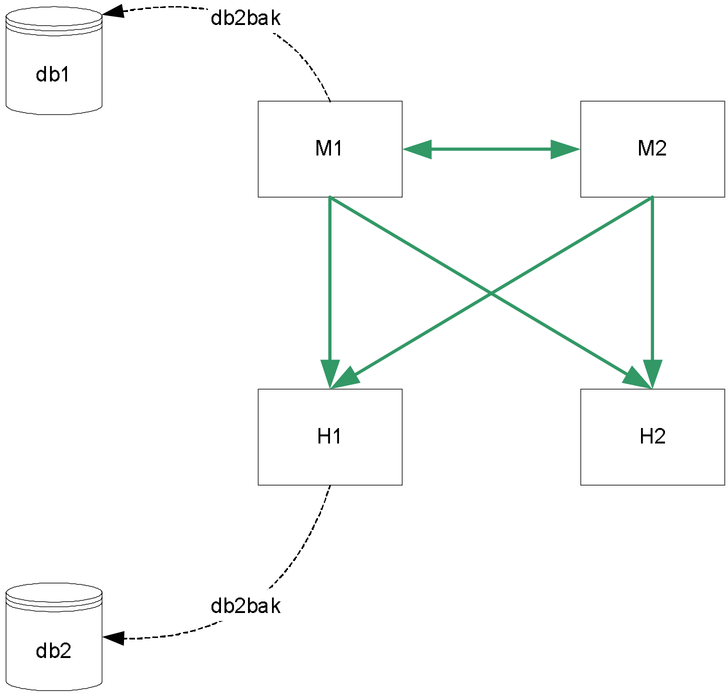
NOTE Because it is easier to restore from a local backup, you should perform a binary backup on each server.

In all of the diagrams that follow in this chapter:

- M = master
- H = hub
- C = consumer
- RA = replication agreement.

Figure 9-1 assumes that M1 and M2 have an identical configuration and that H1 and H2 have an identical configuration. In this scenario, a binary backup would be performed on M1 and on H1. In the case of failure, either master could be restored from the binary backup of M1 (db1) and either hub could be restored from the binary backup of H1 (db2). A master could not be restored from the binary backup of H1 and a hub could not be restored from the binary backup of M1.

Figure 9-1 Binary Backup



Backup to LDIF (db2ldif)

Backup to LDIF is performed at the suffix level. The output of `db2ldif` is a formatted LDIF file. As such, this process takes longer than a binary backup.

NOTES Replication information is not backed up unless you use the `-r` option when running `db2ldif`.

The `dse.ldif` configuration file is not backed up in a backup to LDIF. You should back this file up manually to enable you to restore a previous configuration.

Backup to LDIF has the following advantages:

- Backup to LDIF can be performed from any server, regardless of its configuration.
- Restoration from a backup to LDIF can be performed on any server, regardless of its configuration (if the `-r` option is used to export replication information.)

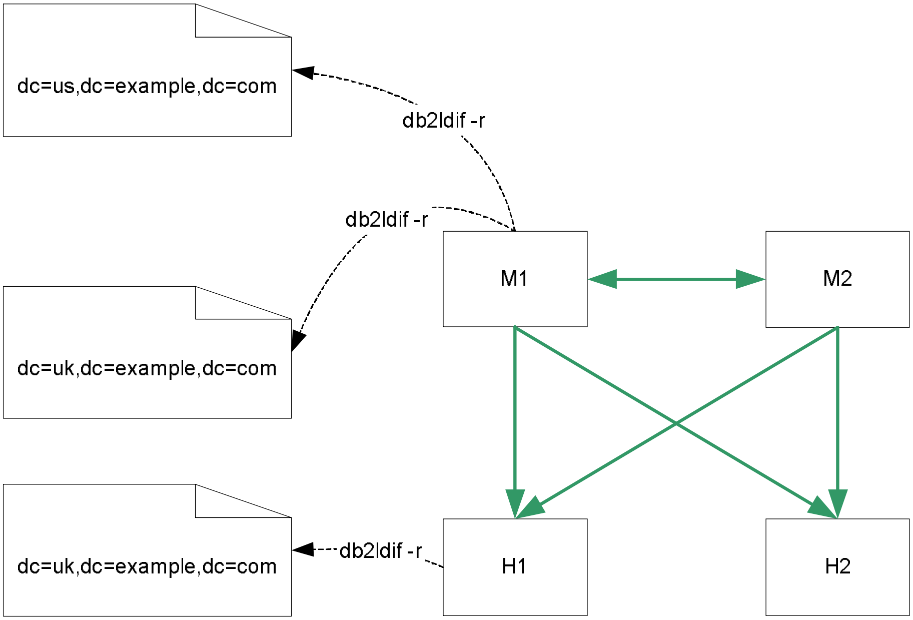
Backing up using backup to LDIF has the following limitations:

- In situations where rapid backup and restoration are required, backup to LDIF may take too long to be viable.

You should perform a regular backup using backup to LDIF for each replicated suffix, on a single master in your topology.

In the following diagram, `db2ldif -r` would be performed for each replicated suffix, on M1 only, or additionally on H1:

Figure 9-2 Backup Using db2ldif -r



CAUTION It is essential that your backup be performed more frequently than the *purge delay*. The purge delay, specified by the `nsDS5ReplicaPurgeDelay` attribute, is the period of time, in seconds, after which internal purge operations are performed on the change log. The default purge delay is 604800 seconds (1 week.) The change log maintains a record of updates, which may or may not have been replicated.

If your backup is less frequent than the purge delay, the change log may be cleared before it has been backed up. Changes will therefore be lost if you use the backup to restore data.

Choosing a Restoration Method

Directory Server provides two methods of restoring data: binary restore (`bak2db`) and restoration from an LDIF file (`ldif2db`). As with the backup methods discussed previously, both of these methods have advantages and limitations.

Binary Restore (bak2db)

Binary restore copies data at the database level. Restoring data using binary restore therefore has the following advantages:

- All suffixes can be restored at once.
- Binary restore is significantly faster than restoring from an LDIF file.

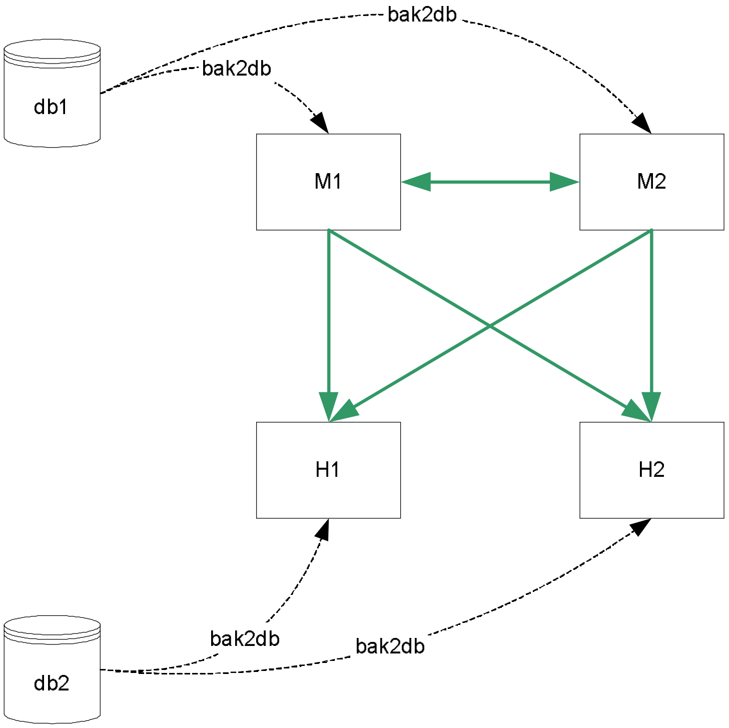
Restoring data using binary restore has the following limitations:

- Restoration can be performed only on a server with an identical configuration, as defined in "Binary Backup (db2bak)" on page 199. For more information on restoring data with the binary restore feature, refer to Initializing a Replica Using Binary Copy in Chapter 8 of the *Directory Server Administration Guide*.
- If you are unaware that your database was corrupt when you performed the binary backup, you risk restoring a corrupt database, since binary backup creates an exact copy of the database.

Binary restore is the preferred restoration method if the machines have an identical configuration, and time is a major consideration.

Figure 9-3 assumes that M1 and M2 have an identical configuration and that H1 and H2 have an identical configuration. In this scenario, either master can be restored from the binary backup of M1 (db1) and either hub can be restored from the binary backup of H1 (db2).

Figure 9-3 Binary Restore



Restoration From LDIF (ldif2db)

Restoration from an LDIF file is performed at the suffix level. As such, this process takes longer than a binary restore. Restoration from an LDIF file has the following advantages:

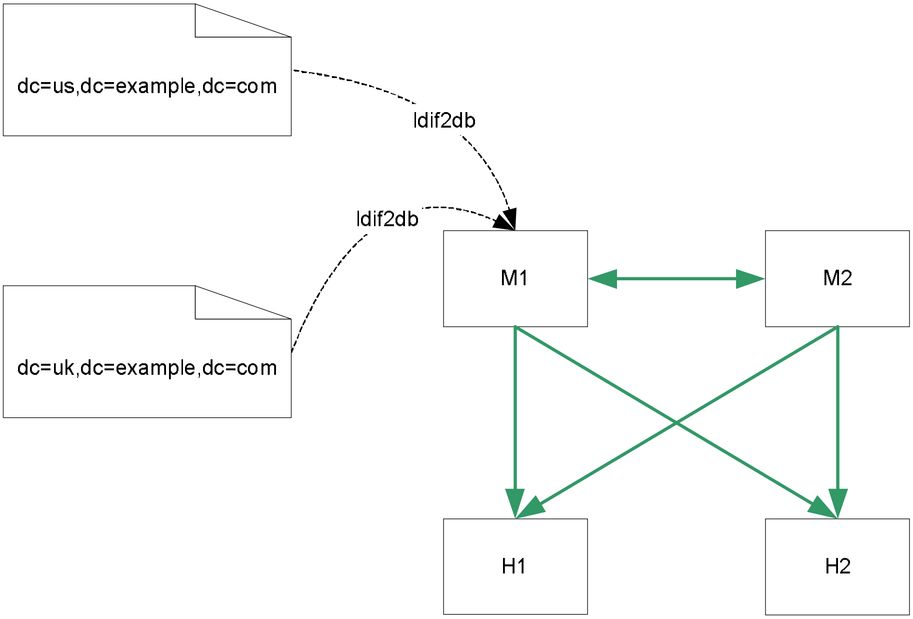
- It can be performed on any server, regardless of its configuration.
- A single LDIF file can be used to deploy an entire directory service, regardless of its replication topology. This is particularly useful for the dynamic expansion and contraction of a directory service according to anticipated business needs.

Restoration from an LDIF file has the following limitations:

- In situations where rapid restoration is required, `ldif2db` may take too long to be viable.

In the following diagram, `ldif2db` can be performed for each replicated suffix, on M1 only, or additionally on H1:

Figure 9-4 Restore Using ldif2db



Sample Replication Topologies

Your replication topology will be determined by the size of your enterprise and the physical location of your data centers. For this reason, we have provided sample replication topologies, categorized by the number of data centers (sites) in which the organization has a directory.

When you first deploy your directory, you will deploy according to the current number of entries and the current volume of reads / writes to the directory. As the number of entries increases, you will need to scale your directory for improved read performance. Suggestions for scalability are provided for each organization.

These topologies aim to provide continued service in the event of failure of one component, without immediate manual intervention. In the case of one and two data centers, local read / write failover is also provided.

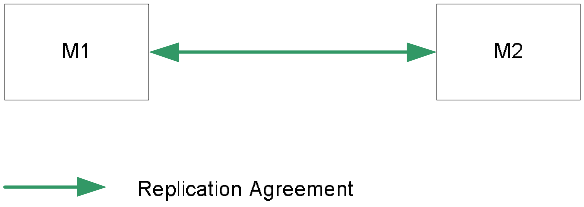
Single Data Center

In a single data center, your topology is largely dependent on the anticipated performance requirements of the directory. In the basic topology suggested, it is assumed that the deployment warrants at least two servers to handle read and write operations. Two masters also provide a high-availability solution.

Single Data Center Basic Topology

The topology depicted in Figure 9-5 assumes one data center, with two masters for read and write performance. In this basic scenario, clients write to either master, and read from either master.

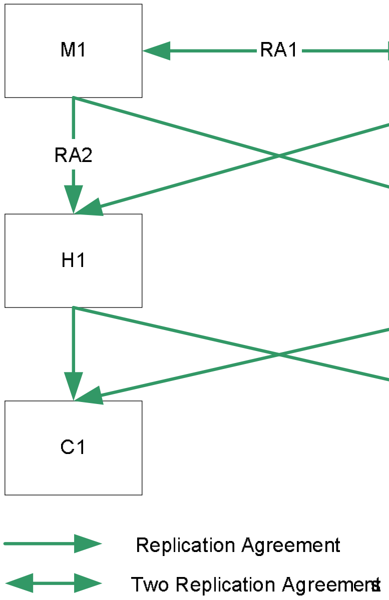
Figure 9-5 One Data Center - Basic Topology



Single Data Center Scaled For Read Performance

Increased read performance is achieved by adding hubs, and then consumers, as indicated in Figure 9-6. Hubs are added below the masters first to make adding a third level of consumers easier. Configuring the second level servers as hubs immediately will allow consumers to be added below them without having to reconfigure any of the machines.

Figure 9-6 One Data Center Scaled For Read Performance



Single Data Center Failure Matrix

In the scenario depicted in Figure 9-6, various components may be rendered unavailable for any one of the reasons described in "Addressing Failure and Recovery" on page 198. These points of failure, and the related recovery actions are described in table Table 9-1.

Table 9-1 Single Data Center - Failure Matrix

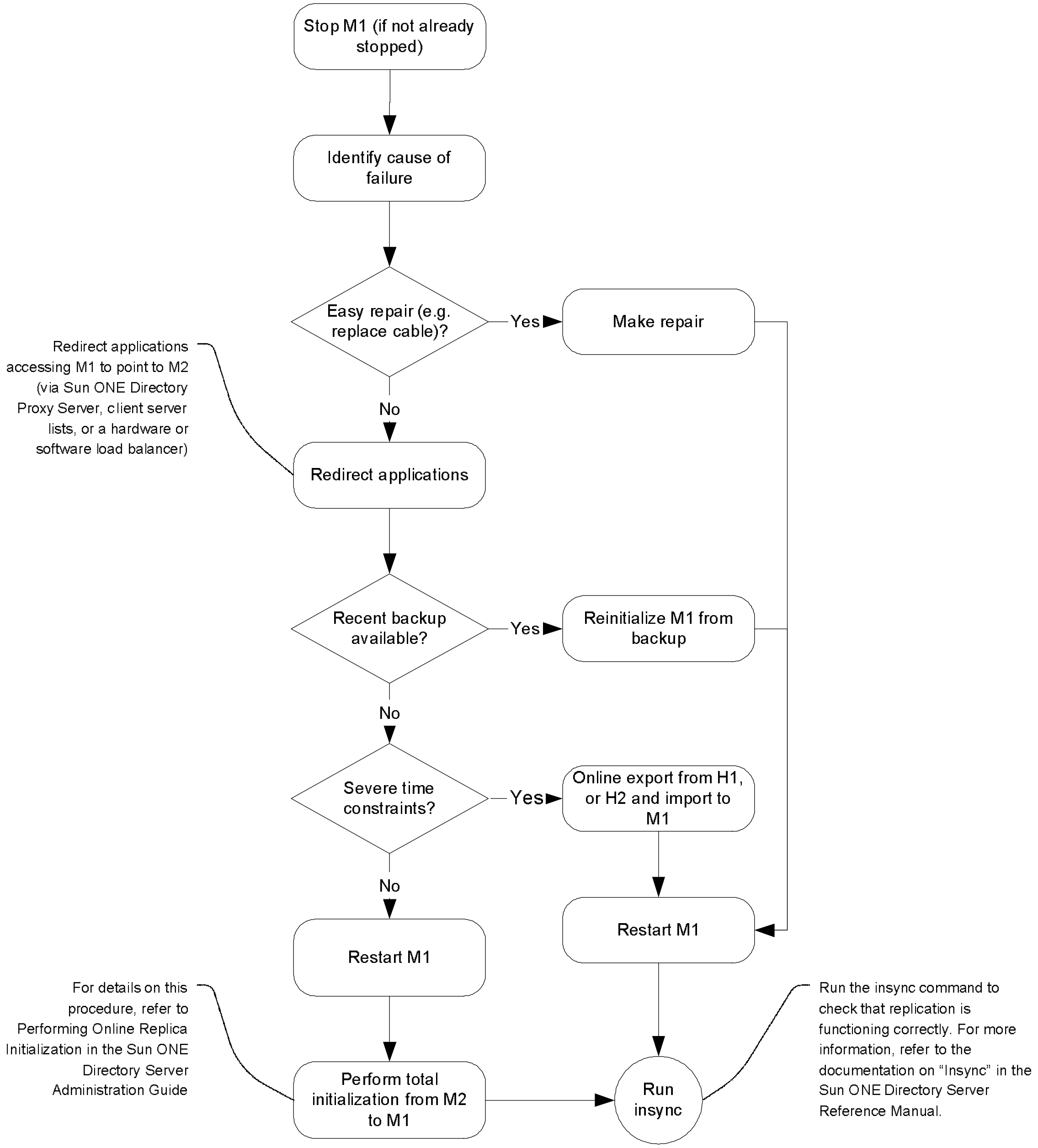
Failed Component	Action
M1	Local writes are routed to M2, via Directory Proxy Server, client server lists, or a hardware or software load balancer. M2 continues to replicate to both H1 and H2.
M2	Local writes are routed to M1, via Directory Proxy Server, client server lists, or a hardware or software load balancer. M1 continues to replicate to both H1 and H2.
LAN link supporting RA1	Both masters continue to receive local writes. Conflicts are resolved at the level of the hubs, assuring that consumers contain the same data.
H1 or H2	Both masters continue to receive local writes. Conflicts are resolved at the level of the masters, and replicated through the alternate hub to all consumers, assuring that consumers contain the same data .
LAN link supporting RA2	Both masters continue to receive local writes. M2 replicates to H1 and replication traffic from the hubs to the consumers continues as normal.

Single Data Center Recovery Procedure (One Component)

In a single data center with two masters, read and write capability is maintained if one master fails. This section describes a sample recovery strategy that can be applied to reinstate the failed component.

The flowchart depicted in Figure 9-7, and the procedure that follows, assumes that one master (M1) has failed.

Figure 9-7 Single Data Center Recovery Sample Procedure (One Component)


Stop M1 (if not already stopped)
Identify cause of failure
Easy repair (e.g. replace cable)?
Yes
Make repair
No
Redirect applications accessing M1 to point to M2 (via Sun ONE Directory Proxy Server, client server lists, or a hardware or software load balancer)
Redirect applications
Recent backup available?
Yes
Reinitialize M1 from backup
No
Severe time constraints?
Yes
Online export from H1, or H2 and import to M1
No
Restart M1
Restart M1
For details on this procedure, refer to Performing Online Replica Initialization in the Sun ONE Directory Server Administration Guide
Perform total initialization from M2 to M1
Run insync
Run the insync command to check that replication is functioning correctly. For more information, refer to the documentation on "Insync" in the Sun ONE Directory Server Reference Manual.

1. Stop M1 (if it is not already stopped).
2. Identify the cause of the failure. If it is easily repaired (by replacing a network cable, for example) make the repair.
3. If the problem is more serious and will take time to fix, ensure that any applications accessing M1 are redirected to point to M2, via Sun Java System Directory Proxy Server, client server lists, or a hardware or software load balancer.
4. If a recent backup is available, re-initialize M1 from the backup.
5. If a recent backup is *not* available, restart M1 and perform a total initialization from M2 to M1. For details on this procedure, refer to Performing Online Replica Initialization in Chapter 8 of the *Directory Server Administration Guide*.
6. If a recent backup is *not* available, and time considerations prevent you from performing a total initialization, perform an online export from H1, or H2, and an import (`ldif2db`) to M1.
7. Start M1 (if it is not already started.)
8. Set M1 to read / write mode (if it is in read-only mode.)
9. Use the `insync` command to check that replication is now functioning correctly. For more information, refer to the documentation on “Insync” in Chapter 1 of the *Directory Server Administration Reference.*

NOTE Performing an online export will impact the performance of the server. You should therefore use a hub for the export, rather than the master, M2, which is currently the only server available for write operations.

Single Data Center Recovery Procedure (Two Components)

In the event of two masters failing in this scenario, write capability is lost. If the failure is serious and will take a long time to repair, it is necessary to implement a strategy that will provide write capability as rapidly as possible.

The following procedure assumes that both M1 and M2 have failed, and are unrecoverable in the near term. Note that you need to assess the quickest and least complicated method of recovery. This procedure depicts server promotion as the least complicated method, for illustration purposes.

1. Promote H1 to a writable master. For information on how to do this, refer to “Promoting or Demoting Replicas” in Chapter 8 of the *Directory Server Administration Guide*.
2. Ensure that any applications that were accessing either M1 or M2 are redirected to point to the new master.
3. Add a replication agreement between the new master and H2 to ensure that modifications continue to be replicated to the consumers.

Two Data Centers

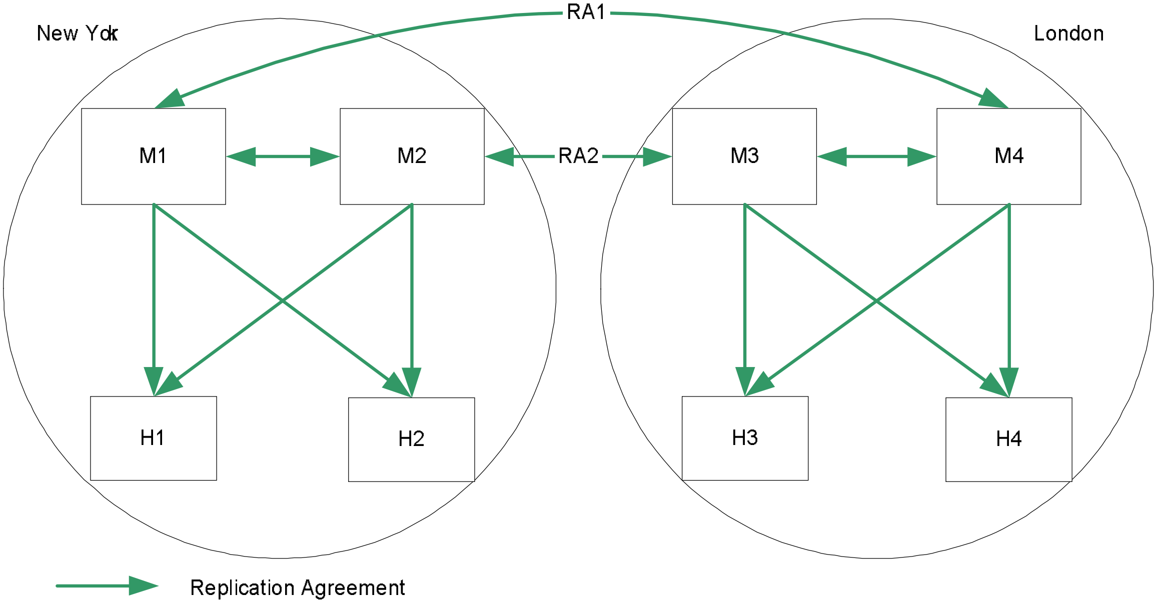
When data is shared across sites, an effective replication topology is imperative, for both performance and failover.

Two Data Centers Basic Topology

The topology depicted in Figure 9-8 assumes two masters and two hubs in each data center, for optimized read and write performance. Configuring the second level servers as hubs immediately will allow consumers to be added below them without having to reconfigure any of the machines.

In this scenario, the replication agreements RA1 and RA2 are configured over *separate* network links. This configuration will enable replication to continue across data centers, in the case of one of the network links becoming unavailable or unreliable.

Figure 9-8 Two Data Centers Basic Topology

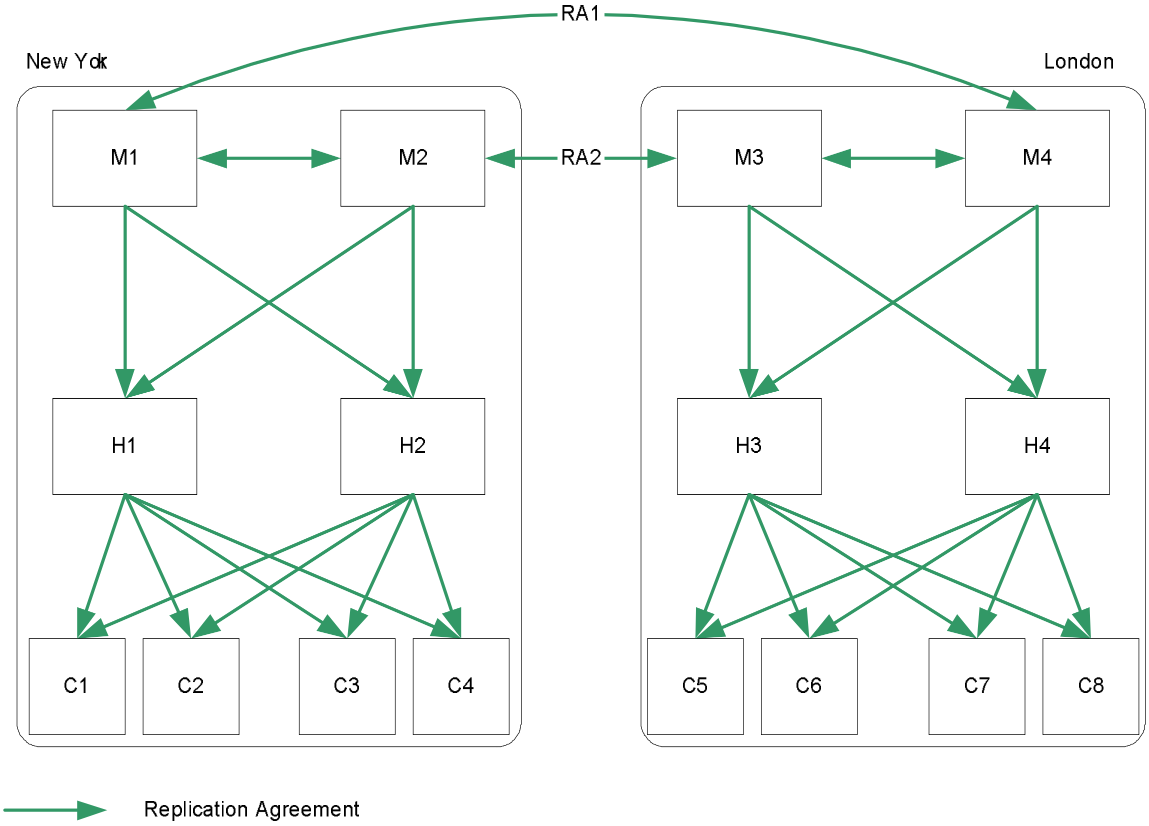


Two Data Centers Scaled For Read Performance

As in the scenario for one data center, increased read performance is achieved by adding hubs, and then consumers, as indicated in Figure 9-6.

In this scenario, the replication agreements RA1 and RA2 are configured over *separate* network links. This configuration will enable replication to continue across data centers, in the case of one of the network links becoming unavailable or unreliable.

Figure 9-9 Two Data Centers Scaled For Read Performance



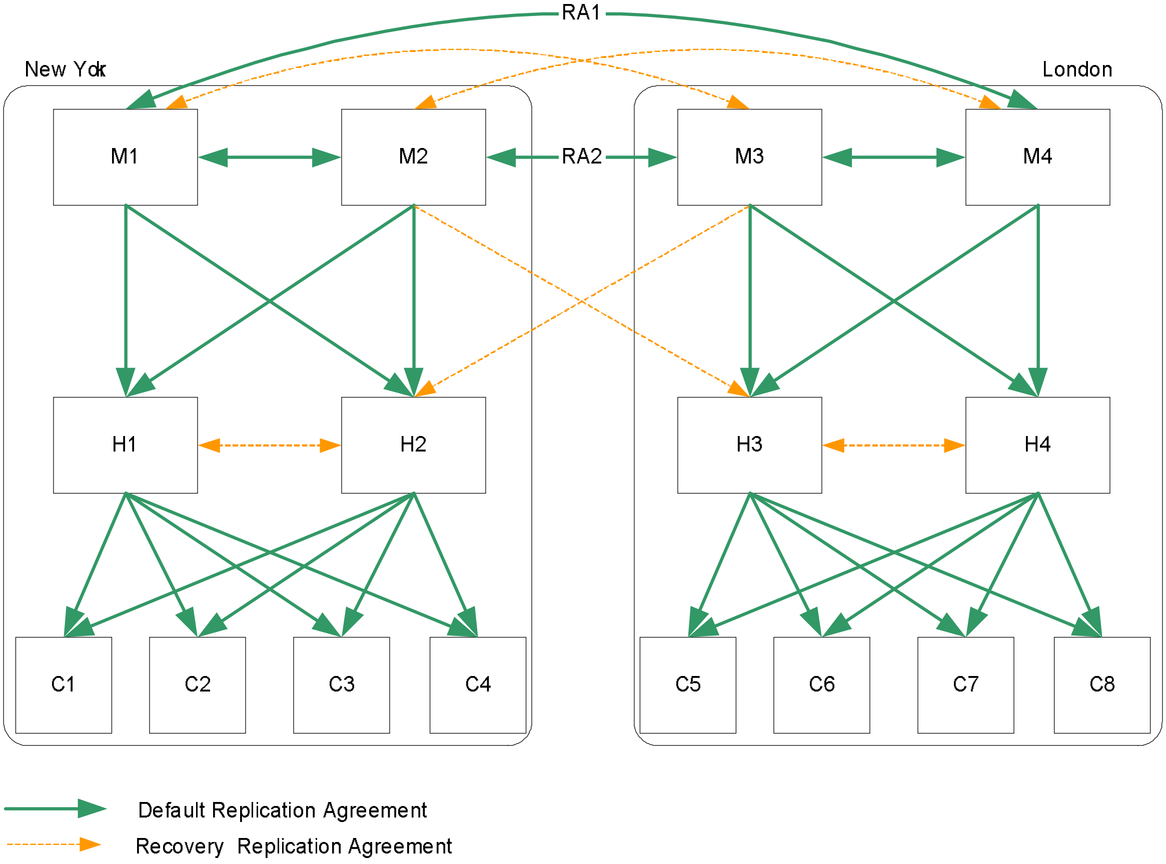
Two Data Centers Recovery Scenarios

For the deployment depicted in Figure 9-9, if one master fails, the same recovery strategy can be applied as described for a single data center. The replication agreements between M1 and M4, and between M2 and M3, will ensure that both data centers continue to receive replicated updates, even if one of the masters in the data center is not available.

If more than one master fails, however, an advanced recovery strategy is required. This involves the creation of recovery replication agreements, that are disabled by default but can be enabled rapidly in the event of a failure.

This recovery strategy is illustrated in Figure 9-10.

Figure 9-10 Two Data Centers Recovery Replication Agreements



The recovery strategy applied will depend on which combination of components fails. However, once you have a basic strategy in place to cope with multiple failures, you can apply that strategy should other components fail.

In the sample topology depicted in Figure 9-10, assume that both masters in the New York data center fail. The recovery strategy in this scenario would be as follows:

1. Enable the recovery replication agreement between M3 and H2.

   This ensures that remote writes on the London site continue to be replicated to the New York site.

2. Promote H2 to a writable master. For information on how to do this, refer to "Promoting or Demoting Replicas" in Chapter 8 of the *Directory Server Administration Guide*.

   This ensures that write-capability is maintained on the New York site.

3. Create a replication agreement between the new promoted master (was H2) and M3.

   This ensures that writes on the New York site continue to be replicated to the London site.

4. Enable the recovery replication agreement between H2 and H1 (one direction only.)

   This ensures that H1 continues to receive updates from the entire replication topology.

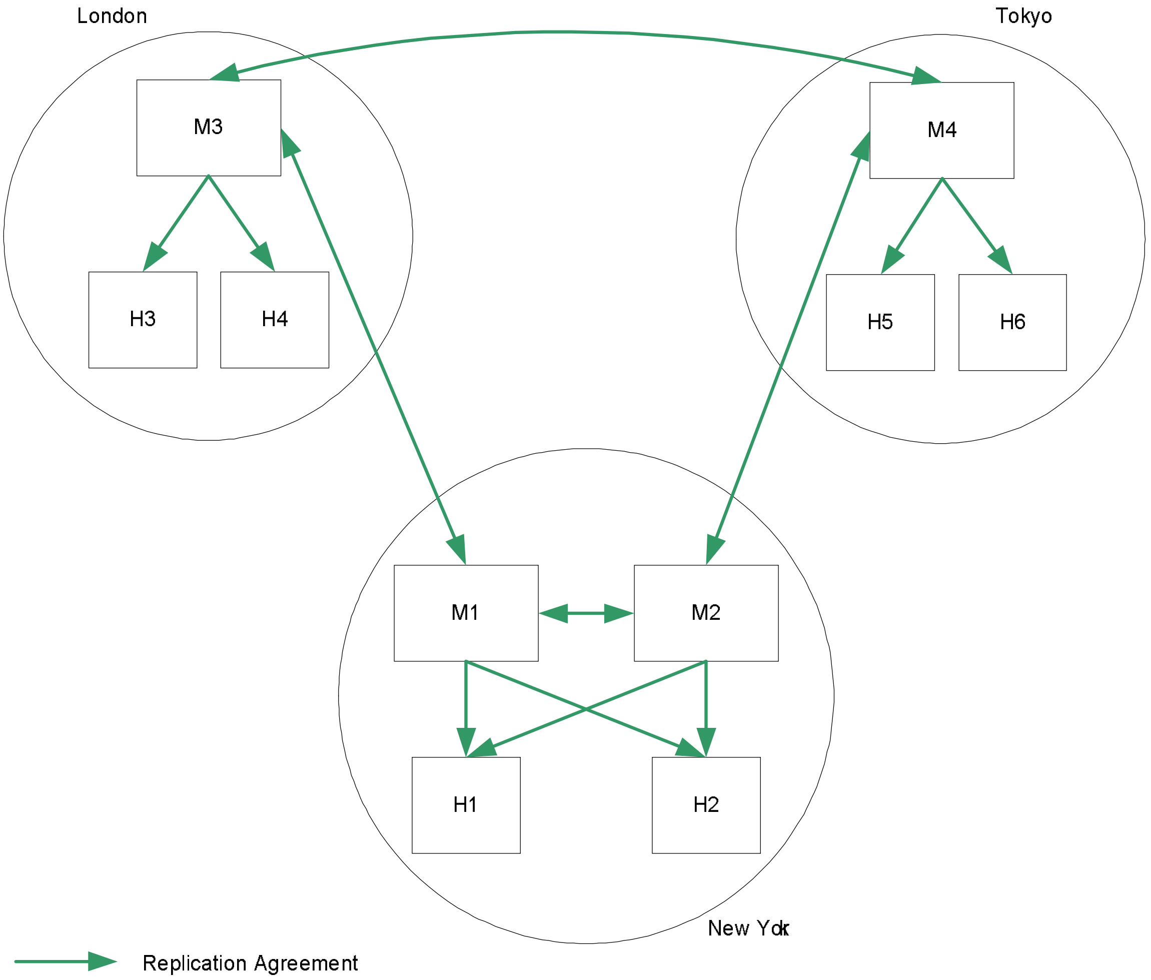
Three Data Centers

Directory Server 5.2 supports four-way multi-master replication. In an enterprise spread over three main geographical regions, you have the possibility of two masters in one data center and one in each of the others. How you divide this directory capacity will be determined (amongst other issues) by the relative volume of read and write traffic anticipated in each data center.

Three Data Centers Basic Topology

The topology depicted in Figure 9-11 assumes that the New York data center receives the largest number of read and write requests, although local read and write requests are possible in each of the three data centers.

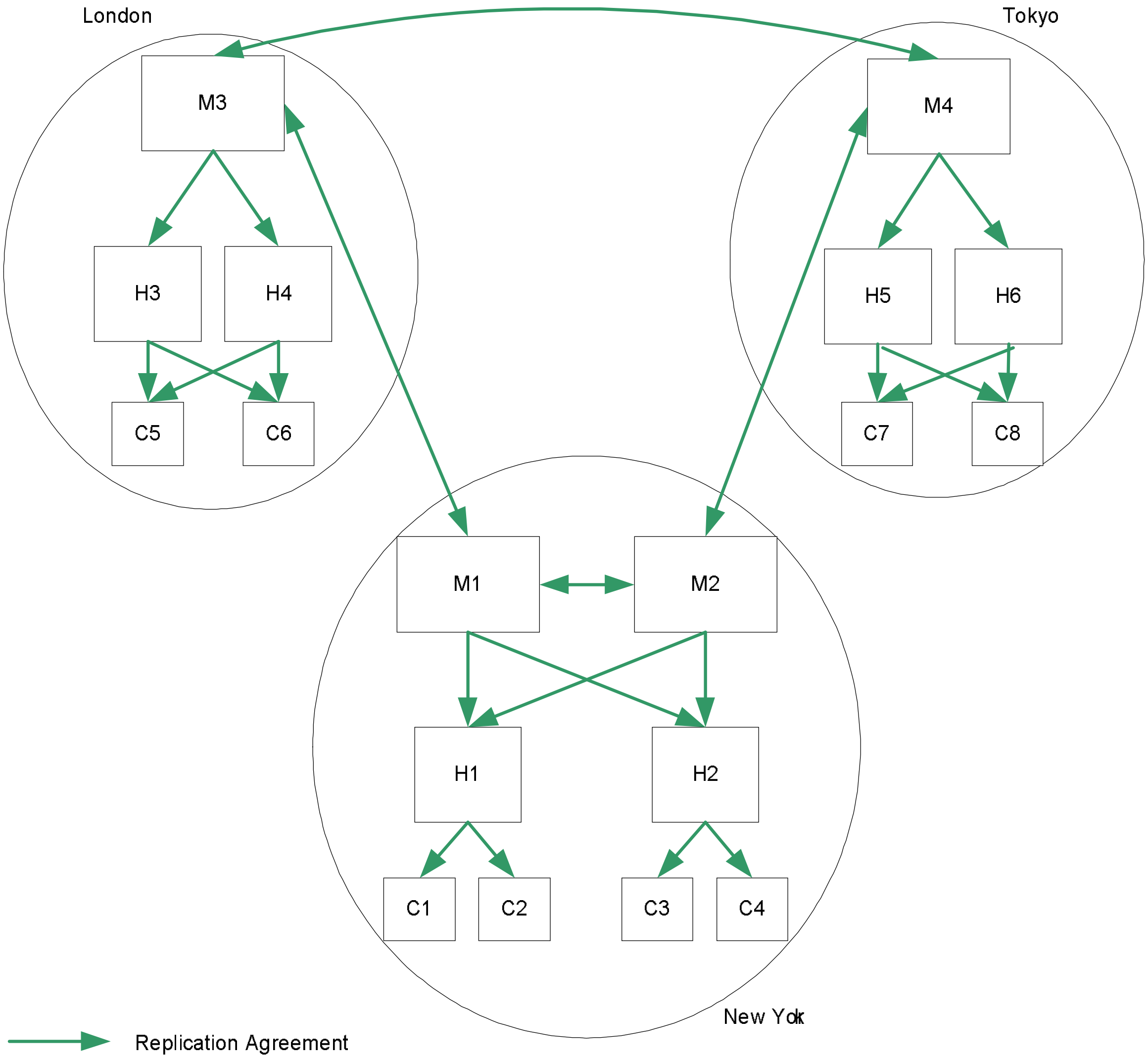
Figure 9-11 Three Data Centers Basic Topology



Three Data Centers Scaled For Read Performance

As in the previous scenarios, increased read performance is achieved by adding hubs and consumers, once again taking into account the anticipated performance requirements across the different data centers. This topology is indicated in Figure 9-12.

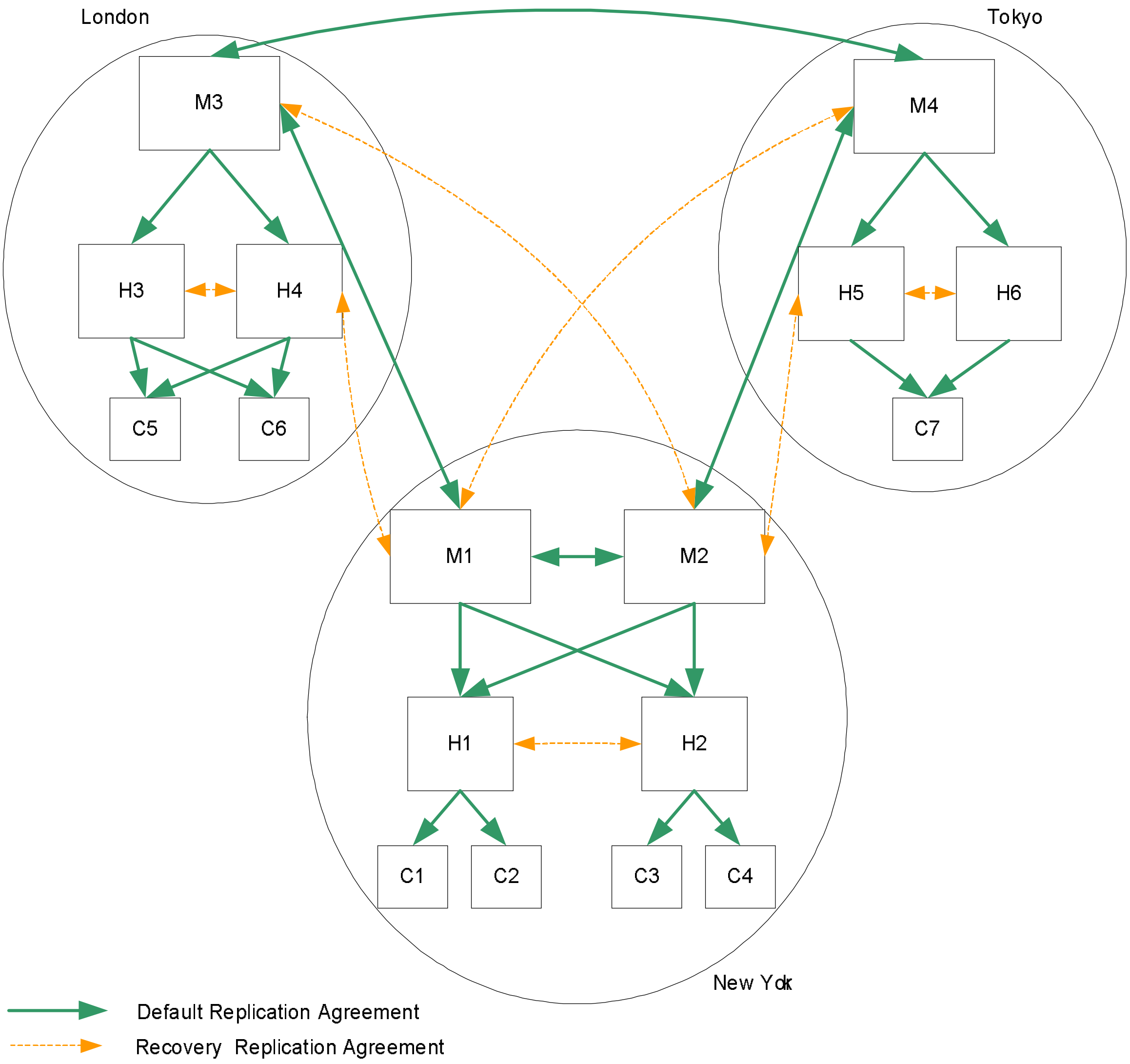
Figure 9-12 Three Data Centers Scaled For Read Performance



Three Data Centers Recovery Scenarios

As was the case for two data centers, if more than one master fails, a recovery strategy involving the creation of recovery replication agreements is required. These agreements are disabled by default but can be enabled rapidly in the event of a failure, as shown in Figure 9-13.

Figure 9-13 Three Data Centers Recovery Replication Agreements



Three Data Centers Recovery Procedure (One Component)

In the scenario depicted in Figure 9-13, losing one master in either London or Tokyo implies that local write capability is lost. The following procedure assumes that M3 (London) has failed.

1. Promote H4 to a writable master. For information on how to do this, refer to “Promoting or Demoting Replicas” in Chapter 8 of the *Directory Server Administration Guide*.
2. Enable the recovery replication agreement from H4 to H3, to ensure that local writes are replicated to all local consumers.
3. Enable the recovery replication agreements between M1 and H4 to ensure that local writes are replicated to remote data centers and that remote writes are replicated to local consumers.
4. Ensure that any applications that were accessing M3 are redirected to point to the new master.

NOTE This procedure is an intermediate solution that will provide immediate local read and write capability, while you set about repairing M3.

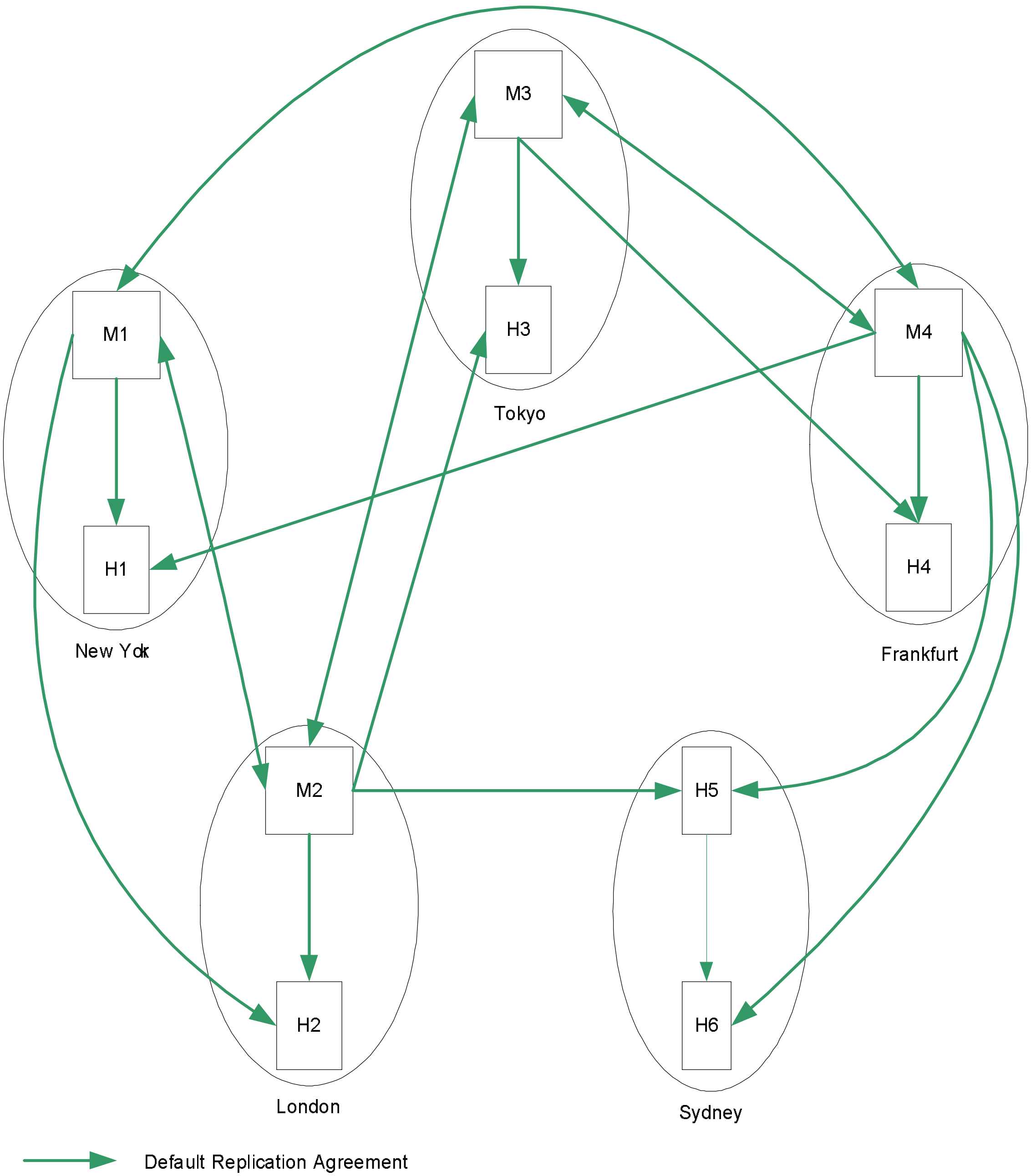
Five Data Centers

Directory Server 5.2 supports four-way multi-master replication. In an enterprise spread over five main geographical regions, you must assess which region has the lowest requirements in terms of local update performance. This region will not have a master server and will redirect writes to one of the masters in the other regions.

Five Data Centers Basic Topology

The topology depicted in Figure 9-14 assumes that the Sydney data center receives the smallest number of write requests. Local read requests are possible in each of the five data centers.

Figure 9-14 Five Data Centers Basic Topology



Five Data Centers Scaled For Read Performance

As in the previous scenarios, increased read performance is achieved by adding hubs and consumers, once again taking into account the anticipated performance requirements across the different data centers.

Five Data Centers Recovery Scenarios

As was the case for two data centers, if more than one master fails, a recovery strategy involving the creation of recovery replication agreements is required. These agreements are disabled by default but can be enabled rapidly in the event of a failure, as shown in Figure 9-15 on page 223.

Five Data Centers Recovery Procedure (One Component)

In the scenario depicted in Figure 9-15, losing a master in any data center implies that local write capability is lost. The following procedure assumes that M1 (New York) has failed.

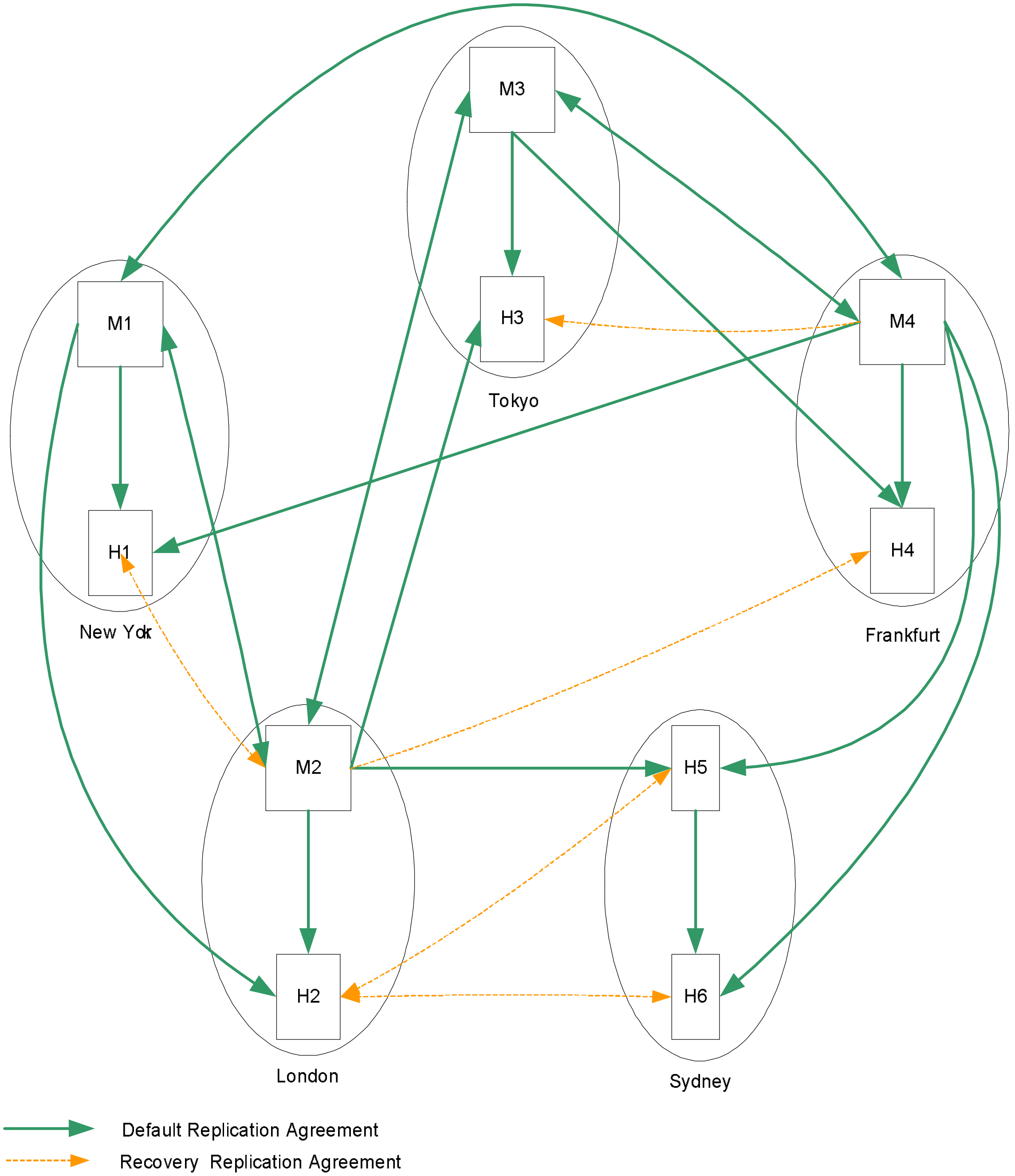
1. Promote H1 to a writable master. For information on how to do this, refer to "Promoting or Demoting Replicas" in Chapter 8 of the *Directory Server Administration Guide*.

2. Enable the recovery replication agreement from M2 to H1, to ensure that local writes are replicated to remote data centers and that remote writes are replicated to local consumers.

3. Ensure that any applications that were accessing M1 are redirected to point to the new master.

NOTE This procedure is an intermediate solution that will provide immediate local read and write capability, while you set about repairing M1.

Figure 9-15 Five Data Centers Recovery Replication Agreements



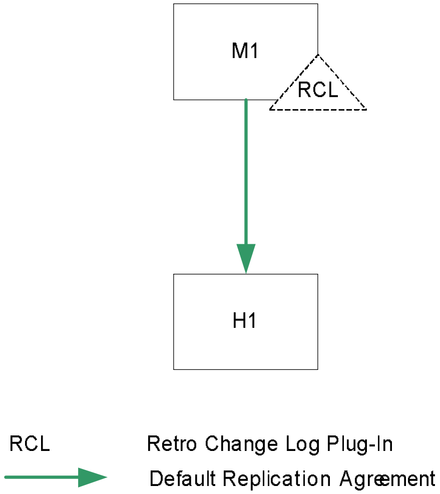
Single Data Center Using the Retro Change Log Plug-In

The previous topology for a single data center does not take into account the use of the retro change log plug-in. Most applications that rely on the retro change log assume that it contains ordered changes and thus would fail due to the effects on the retro change log of the loose consistency multi-master replication model. In general, if the requirements of any application being deployed include the retro change log, a multi-master replication topology should not be used in this deployment. For more information, refer to "Replication and the Retro Change Log Plug-In" on page 137, and to Using the Retro Change Log Plug-In in Chapter 8 of the *Directory Server Administration Guide*.

Retro Change Log Plug-in Basic Topology

If multi-master replication cannot be deployed, the basic topology depicted in Figure 9-16 is suggested.

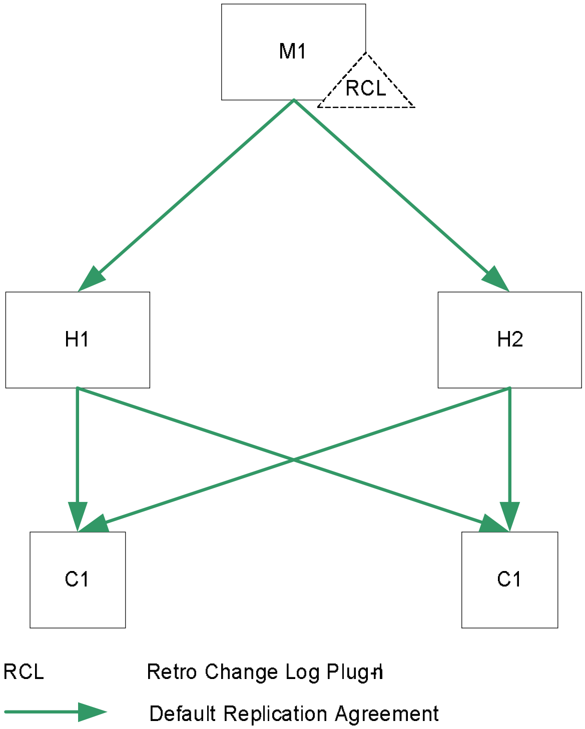
Figure 9-16 One Data Center Using the Retro Change Log Plug-in



Retro Change Log Plug-in Scaled For Read Performance

As in the standard single data center topology, increased read performance is achieved by adding hubs, and then consumers, as indicated in Figure 9-17.

Figure 9-17 One Data Center Using the Retro Change Log Plug-in (Scaled)



Retro Change Log Plug-in Recovery Procedure

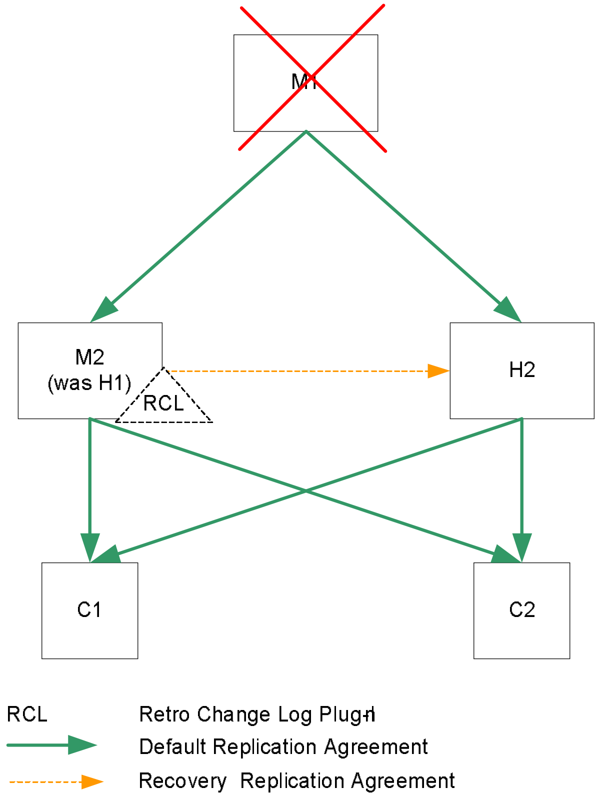
For the deployment depicted in Figure 9-17, the following strategy can be applied if the master server fails:

1. Stop M1 (if it is not already stopped).
2. Promote H1 or H2 to a master server. For information on how to do this, refer to “Promoting or Demoting Replicas” in Chapter 8 of the *Directory Server Administration Guide*.
3. Enable the retro change log plug-in on the new master (M2.)

4. Restore the backup retro change log on M2.
5. Restart the server.
6. Add a replication agreement between M2 and the remaining hub to ensure that modifications continue to be replicated to the hub.

   This recovery strategy is illustrated in Figure 9-18.

Figure 9-18 One Data Center Using the Retro Change Log Plug-in (Recovery)



Chapter 10

System Sizing

Appropriate hardware sizing is a critical component of directory service planning and deployment. When sizing hardware, the amount of memory available and the amount of local disk space available are of key importance.

NOTE For best results, install and configure a test system with a subset of entries representing those used in production. You can then use the test system to approximate the behavior of the production server.

When optimizing for particular systems, ensure you understand how system buses, peripheral buses, I/O devices, and supported file systems work so you can take advantage of I/O subsystem features when tuning these to support Directory Server.

This chapter suggests ways of estimating disk and memory requirements for a Directory Server instance. It also touches on network and SSL accelerator hardware requirements.

Suggested Minimum Requirements

Table 10-1 proposes minimum memory and disk space requirements for installing and using the software in a production environment.

Minimum requirements for specified numbers of entries may in fact differ from those provided in Table 10-1. Sizes here reflect relatively small entries, with indexes set according to the default configuration, and with caches minimally tuned. If entries include large binary attribute values such as digital photos, or if indexing or caching is configured differently, then revise minimum disk space and memory estimates upward accordingly.

Table 10-1 Minimum Disk Space and Memory Requirements

Required for...	Free Local Disk Space	Free RAM
Unpacking product	At least 125 MB	-
Product installation	At least 200 MB	At least 256 MB
10,000-250,000 entries	Add at least 3 GB	Add at least 256 MB
250,000-1,000,000 entries	Add at least 5 GB	Add at least 512 MB
Over 1,000,000 entries	Add 8 GB or more	Add 1 GB or more

Minimum disk space requirements include 1 GB devoted to access logs. By default, Directory Server is configured to rotate through 10 access log files (`nsslapd-accesslog-maxlogsperdir` on `cn=config`) each holding up to 100 MB (`nsslapd-accesslog-maxlogsize` on `cn=config`) of messages. Volume for error and audit logs depends on how Directory Server is configured. Refer to the "Monitoring Directory Server Using Log Files,“ in the *Directory Server Administration Guide* for details on configuring logging.

Minimum Available Memory

Minimum memory estimates reflect memory used by an instance of Directory Server in a typical deployment. The estimates do not account for memory used by the system and by other applications. For a more accurate picture, you must measure memory use empirically. Refer to "Sizing Physical Memory" on page 229 for details.

As a rule, the more available memory, the better.

Minimum Local Disk Space

Minimum local disk space estimates reflect the space needed for an instance of Directory Server in a typical deployment. Experience suggests that if directory entries are large, the space needed is at minimum four times the size of the equivalent LDIF on disk. Refer to "Sizing Disk Subsystems" on page 233 for details.

Do *not* install the server or any data it accesses on network disks. Directory Server software does not support the use of network attached storage via NFS, AFS, or SMB. Instead, all configuration, log, database, and index files must reside on local storage at all times, even after installation.

Minimum Processing Power

High volume systems typically employ multiple, high-speed processors to provide appropriate processing power for multiple simultaneous searches, extensive indexing, replication, and other features. Refer to “Sizing for Multiprocessor Systems” on page 242 for details.

Minimum Network Capacity

Testing has demonstrated that 100 Mbit Ethernet may be sufficient for even service provider performance, depending on the maximum throughput expected. You may estimate theoretical maximum throughput as follows:

```
max. throughput = max. entries returned/second x average entry size
```

Imagine for example that a Directory Server must respond to a peak of 5000 searches per second for which it returns 1 entry each with entries having average size of 2000 bytes, then the theoretical maximum throughput would be 10 MB, or 80 Mbit. 80 Mbit is likely to be more than a single 100 Mbit Ethernet adapter can provide. Actual observed performance may vary.

If you expect to perform multi-master replication over a wide area network, ensure the connection provides sufficient throughput with minimum latency and near-zero packet loss.

Refer to “Sizing Network Capacity” on page 243 for more information.

Sizing Physical Memory

Directory Server stores information using database technology. As is the case for any application relying on database technology, adequate fast memory is key to optimum Directory Server performance. As a rule, the more memory available, the more directory information can be cached for quick access. In the ideal case, each server has enough memory to cache the entire contents of the directory at all times. As Directory Server 5.2 supports 64-bit memory addressing, it is now possible to handle total cache sizes of as much as the 64-bit processor can address.

NOTE When deploying Directory Server in a production environment, configure cache sizes well below theoretical process limits, leaving appropriate resources available for general system operation.

Estimating memory size required to run Directory Server involves estimating the memory needed both for a specific Directory Server configuration, and for the underlying system on which Directory Server runs.

Sizing Memory for Directory Server

Given estimated configuration values for a specific deployment, you can estimate physical memory needed for an instance of Directory Server. Table 10-2 summarizes the values used for the calculations in this section.

Table 10-2 Values for Sizing Memory for Directory Server

Value	Description[1]
`nsslapd-cachememsize`	Entry cache size for a suffix An entry cache contains formatted entries, ready to be sent in response to a client request. One instance may handle several entry caches.
`nsslapd-dbcachesize`	Database cache size The database cache holds elements from databases and indexes used by the server.
`nsslapd-import-cachesize`	Database cache size for bulk import Import cache is used only when importing entries. You may be able to avoid budgeting extra memory for import cache, instead reusing memory budgeted for entry or database cache if you perform *only* offline imports.
`nsslapd-maxconnections`	Maximum number of connections managed.
`nsslapd-threadnumber`	Number of operation threads created at server startup

[1] For complete descriptions, refer to the *Directory Server Administration Reference*.

To estimate approximate memory size, perform the following steps.

1. Estimate the base size of the server process, `slapdBase`.

```
slapdBase     = 75 MB +(nsslapd-threadnumber x 0.5 MB) +(nsslapd-maxconnections x 0.5 KB)
```

2. Determine the sum of entry cache sizes, `entryCacheSum`.

$\texttt{entryCacheSum = Sum}_{\texttt{all entry caches}}\texttt{(nsslapd-cachememsize)}$

Note that the entry cache includes an allocation overhead (in other words, the cache will consume more memory than you specify in the `nsslapd-cachememsize` parameter.) This may appear to be a memory leak, but it is not. Depending on how the memory allocation library handles requests, actual memory used may be much larger than the memory specified. For more information see “Entry Cache,” in Chapter 3 of the *Directory Server Performance Tuning Guide*.

3. Determine the total size for all caches, `cacheSum`.

```
cacheSum      = entryCacheSum + nsslapd-dbcachesize + nsslapd-import-cachesize
```

4. Determine the total size for the Directory Server process, `slapdSize`.

```
slapdSize     = slapdBase + cacheSum
```

   You may use utilities such as `pmap`(1) on Solaris systems or the Windows Task Manager to measure physical memory used by Directory Server.

5. Estimated memory needed to handle incoming client requests, `slapdGrowth`.

```
slapdGrowth   = 20% x slapdSize
```

   As a first estimate, we assume 20 percent overhead for handling client requests. The actual percentage may depend on the characteristics of your particular deployment. Validate this percentage empirically before putting Directory Server into production.

6. Determine total memory size for Directory Server, `slapdTotal`.

```
slapdTotal    = slapdSize + slapdGrowth
```

   For large deployments involving 32-bit servers, `slapdTotal` may exceed the practical limit of about 3.4 GB, (2.5GB on Linux systems) and perhaps even the theoretical process limit of about 3.7 GB. In this case, you may choose either to tune caching to work within the limits of the system, or to use a 64-bit version of the product. For more information, see Chapter 3, “Tuning Cache Sizes,” in the *Directory Server Performance Tuning Guide*.

Sizing Memory for the Operating System

Estimating the memory needed to run the underlying operating system must be done empirically, as operating system memory requirements vary widely based on the specifics of the system configuration. For this reason, consider tuning a representative system for deployment before attempting to estimate how much memory the underlying operating system needs. For more information, see

Chapter 2, “Tuning the Operating System,” in the *Directory Server Performance Tuning Guide*. After tuning the system, monitor memory use to arrive at an initial estimate, `systemBase`. You may use utilities such as `sar`(1M) on Solaris systems or the Task Manager on Windows to measure memory use.

NOTE For top performance, dedicate the system running Directory Server to this service only.

If you must run other applications or services, monitor the memory they use as well when sizing total memory required.

Additionally, allocate memory for general system overhead and normal administrative use. A first estimate for this amount, `systemOverhead`, should be at least several hundred megabytes, or 10 percent of the total physical memory, whichever is greater. The goal is to allocate enough space for `systemOverhead` that the system avoids swapping pages in and out of memory while in production.

The total memory needed by the operating system, `systemTotal`, can then be estimated as follows.

```
systemTotal = systemBase + systemOverhead
```

Sizing Total Memory

Given `slapdTotal` and `systemTotal` estimates from the preceding sections, estimate the total memory needed, `totalRAM`.

```
totalRAM = slapdTotal + systemTotal
```

Notice `totalRAM` is an *estimate* of the total memory needed, including the assumption that the system is dedicated to the Directory Server process, and including estimated memory use for all other applications and services expected to run on the system.

Dealing With Insufficient Memory

In many cases, it is not cost effective to provide enough memory to cache all data used by Directory Server.

At minimum, equip the server with enough memory that running Directory Server does not cause constant page swapping. Constant page swapping has a strong negative performance impact. You may use utilities such as `vmstat`(1M) on Solaris and other systems to view memory statistics before and after starting Directory Server and priming the entry cache. Unsupported utilities available separately such as `MemTool` for Solaris systems can be useful in monitoring how memory is used and allocated when applications are running on a test system.

If the system cannot accommodate additional memory, yet you continue to observe constant page swapping, reduce the size of the database and entry caches. Running out of swap space can cause Directory Server to shut itself down.

Refer to Chapter 3, "Tuning Cache Sizes," in the *Directory Server Performance Tuning Guide*, for a discussion of the alternatives available when providing adequate physical memory to cache all directory data is not an option.

Sizing Disk Subsystems

Disk use and I/O capabilities can strongly impact performance. Especially for a deployment supporting large numbers of modifications, the disk subsystem can become an I/O bottleneck. This section offers recommendations for estimating overall disk capacity for a Directory Server instance, and for alleviating disk I/O bottlenecks.

Refer to Chapter 5, "Tuning Logging," in the *Directory Server Performance Tuning Guide*, for more information on alleviating disk I/O bottlenecks.

Sizing Directory Suffixes

Disk space requirements for a suffix depend not only on the size and number of entries in the directory, but also on the directory configuration and in particular how the suffix is indexed. To gauge disk space needed for a large deployment, perform the following steps:

1. Generate LDIF for three representative sets of entries like those expected for deployment, one of 10,000 entries, one of 100,000, one of 1,000,000.

   Generated entries should reflect not only the mix of entry types (users, groups, roles, entries for extended schema) expected, but also the average size of individual attribute values, especially if single large attribute values such as `userCertificate` and `jpegPhoto` are expected.

2. Configure an instance of Directory Server as expected for deployment.

   In particular, index the database as you would for the production directory. If you expect to add indexes later, expect to have to add space for those indexes as well.

3. Load each set of entries, recording the disk space used for each set.

4. Graph the results to extrapolate estimated suffix size for deployment.

5. Add extra disk space to compensate for error and variation.

If you are using replication, note that entry state information (a list of old values) is stored with the entry, and used during conflict resolution. State information can cause an entry to grow significantly in size and should be taken into account when sizing the suffix.

Disk space for suffixes is only part of the picture; you must also consider how Directory Server uses disks.

How Directory Server Uses Disks

Directory suffixes are part of what Directory Server stores on disk. A number of other factors affecting disk use may vary widely depending even on how Directory Server is used after deployment and so are covered here in general terms. Refer to the *Directory Server Administration Guide* for instructions on configuring the items discussed here.

Directory Server Binaries

You need approximately 200 MB disk space to install this version of Directory Server. This estimate is not meant to include space for data or logs, but only for the product binaries.

Event Logging

Disk use estimates for log files depend on the rate of Directory Server activity, the type and level of logging, and the strategy for log rotation.

Many logging requirements can be predicted and planned in advance. If Directory Server writes to logs and in particular audit logs, disk use increases with load level. When high load deployments call for extensive logging, plan for extra disk space to accommodate the high load. You may decrease disk space requirements for deployments with high load logging by establishing an intelligent log rotation and archival system, rotating the logs often, and automating migration of old files to less expensive, higher capacity storage mediums such as tape or cheaper disk clusters.

Some logging requirements cannot easily be predicted. Debug logging can cause temporary but explosive growth in the size of the `errors` log, for example. For a large, high load deployment, consider setting aside several gigabytes of dedicated disk space for temporary, high-volume debug logging. Refer to Chapter 5, “Tuning Logging,” in the *Directory Server Performance Tuning Guide*, for further information.

Transaction Log

Transaction log volume depends upon peak write loads. If writes occur in bursts, transaction logs use more space than if the write load is constant. Directory Server trims transaction logs periodically. Transaction logs therefore should not continue to grow unchecked.

Transaction logs are not flushed during online backup, because database files cannot be modified while they are being copied (this would result in an inconsistent data image.) Transaction logs are copied to the backup location as the last step of the backup.

Directory Server is generally run with durable transactions enabled. When durable transaction capabilities are enabled, Directory Server performs a synchronous write to the transaction log for each modification (`add`, `delete`, `modify`, `modrdn`) operation. In this case, an operation can be blocked if the disk is busy, resulting in a potential I/O bottleneck.

If update performance is critical, plan to use a disk subsystem having fast write cache for the transaction log. Refer to Chapter 5, “Tuning Logging,” in the *Directory Server Performance Tuning Guide*, for further information.

Replication Changelog Database

If the deployment involves replication, the Directory Server suppliers perform change logging. Changelog size depends on the volume of modifications and on the type of changelog trimming employed. Plan capacity based on how the changelog is trimmed. For a large, high load deployment, consider setting aside several gigabytes of disk space to handle changelog growth during periods of abnormally high modification rates. Refer to Chapter 5, “Tuning Logging,” in the *Directory Server Performance Tuning Guide*, for further information.

Suffix Initialization and LDIF Files

During suffix initialization, also called bulk loading or importing, Directory Server requires disk space not only for the suffix database files and the LDIF used to initialize the suffix, but also for intermediate files used during the initialization process. Plan extra (temporary) capacity in the same directory as the database files for the LDIF files and for the intermediate files used during suffix initialization. This may be as much as double the size of the largest index, depending on what indexes you have created.

Backups and LDIF Files

Backups often consume a great deal of disk space. The size of a backup equals the size of the database files involved, and the transaction logs. Accommodate for several backups by allocating space equal to several times the volume of the database files, ensuring that databases and their corresponding backups are maintained on separate disks. Employ intelligent strategies for migrating backups to cheaper storage mediums as they age.

If the deployment involves replication, plan additional space to hold initialization LDIF files, as these differ from backup LDIF files.

Memory Based Rather Than Disk Based File Systems

Some systems support memory based `tmpfs` file systems. On Solaris for example `/tmp` is often mounted as a memory based file system to increase performance.

Only database cache files should be placed on a memory based file system. For more information, see “nsslapd-db-home-directory“ in the *Directory Server Administration Reference*. Never put database or transaction log binaries or configuration files on a memory based file system.

If cache files are placed on `/tmp`, a location shared with other applications on the system, ensure that the system never runs out of space under `/tmp`. Otherwise, when memory is low, Directory Server files in memory based file systems may be paged to the disk space dedicated for the swap partition.

Some systems support RAM disks and other alternative memory based file systems. Refer to the operating system documentation for instructions on creating and administering memory based file systems. Notice that everything in such file systems is volatile and must be reloaded into memory after system reboot. This reinitialization can take a long time to complete, depending on factors such as the processor speed, memory speed, and memory size.

Core Files

Leave room for at minimum one or two `core` files. Although Directory Server should not dump core, recovery and troubleshooting after a crash can be greatly simplified if the `core` file generated during the crash is available for inspection. When generated, `core` files are stored either in the same directory as the file specified by `nsslapd-errorlog` on `cn=config`, or under *ServerRoot*`/bin/slapd/server/` if a crash occurs during startup.

Space for Administration

Leave room for expected system use, including system and Directory Server administration. Ensure that sufficient space is allocated for the base Directory Server installation, for the configuration suffix if it resides on the local instance, for configuration files, and so forth.

Distributing Files Across Disks

By placing commonly-updated Directory Server database and log files on separate disk subsystems, you can spread I/O traffic across multiple disk spindles and controllers, avoiding I/O bottlenecks. Consider providing dedicated disk subsystems for each of the following items.

Transaction Logs

When durable transaction capabilities are enabled, Directory Server performs a synchronous write to the transaction log for each modification operation. An operation is thus blocked when the disk is busy. Placing transaction logs on a dedicated disk can improve write performance, and increase the modification rate Directory Server can handle.

Refer to “Transaction Logging,” in Chapter 5 of the *Directory Server Performance Tuning Guide*.

Databases

Multiple database support allows each database to reside on its own physical disk. You can thus distribute the Directory Server load across multiple databases each on its own disk subsystem. To prevent I/O contention for database operations, consider placing each set of database files on a separate disk subsystem.

For top performance, place database files on a dedicated fast disk subsystem with a large I/O buffer. Directory Server reads data from the disk when it cannot find candidate entries in cache. It regularly flushes writes. Having a fast, dedicated disk subsystem for these operations can alleviate a potential I/O bottleneck.

The `nsslapd-directory` attribute on `cn=config,cn=ldbm database,cn=plugins,cn=config` specifies the disk location where Directory Server stores database files, including index files. By default, such files are located under *ServerRoot*`/slapd-`*ServerID*`/db/`.

Changing database location of course requires not only that you restart Directory Server, but also that you rebuild the database completely. Changing database location on a production server can be a major undertaking, so identify your most important database and put it on a separate disk before putting the server into production.

Log Files

Directory Server provides access, error, and audit logs featuring buffered logging capabilities. Despite buffering, writes to these log files require disk access that may contend with other I/O operations. Consider placing log files on separate disks for improved performance, capacity, and management.

Refer to Chapter 5, “Tuning Logging,” in the *Directory Server Performance Tuning Guide*, for more information.

Cache Files on Memory Based File Systems

In a `tmpfs` file system, for example, files are swapped to disk only when physical memory is exhausted. Given sufficient memory to hold all cache files in physical memory, you may derive improved performance by allocating equivalent disk space for a `tmpfs` file system on Solaris platforms or other memory based file systems such as RAM disks for other platforms, and setting the value of `nsslapd-db-home-directory` to have the Directory Server store cache files on that file system. This prevents the system from unnecessarily flushing memory mapped cache files to disk.

Disk Subsystem Alternatives

“Fast, cheap, safe: pick any two.” — Sun Performance and Tuning, Cockroft and Pettit.

Fast and Safe

When implementing a deployment in which both performance and uptime are critical, consider hardware-based RAID controllers having non-volatile memory caches to provide high speed buffered I/O distributed across large arrays of disks. By spreading load across many spindles and buffering access over very fast connections, I/O can be optimized, and excellent stability provided through high performance RAID striping or parity blocks.

Large non-volatile I/O buffers and high performance disk subsystems such as those offered in Sun StorEdge™ products can greatly enhance Directory Server performance and uptime.

Fast write cache cards provide potential write performance improvements, especially when dedicated for database and/or transaction log use. Fast write cache cards provide non-volatile memory cache that is independent from the disk controller.

Fast and Cheap

For fast, low-cost performance, ensure you have adequate capacity distributed across a number of disks. Consider disks having high rotation speed and low seek times. For best results, dedicate one disk to each distributed component. Consider using multi-master replication to avoid single points of failure.

Cheap and Safe

For cheap, safe configurations, consider low-cost, software-based RAID controllers such as Solaris Volume Manager.

RAID Alternatives

RAID stands for Redundant Array of Inexpensive Disks. As the name suggests, the primary purpose of RAID is to provide resiliency. If one disk in the array fails, data on that disk is not lost but remains available on one or more other disks in the array. To implement resiliency, RAID provides an abstraction allowing multiple disk drives to be configured as a larger virtual disk, usually referred to as a volume. This is achieved by concatenating, mirroring, or striping physical disks. Concatenation is implemented by having blocks of one disk logically follow those of another disk. For example, disk 1 has blocks 0-99, disk 2 has blocks 100-199 and so forth. Mirroring is implemented by copying blocks of one disk to another and then keeping them in continuous synchronization. Striping uses algorithms to distribute virtual disk blocks over multiple physical disks.

The purpose of striping is performance. Random writes can be dealt with very quickly as data being written is likely to be destined for more than one of the disks in the striped volume, hence the disks are able to work in parallel. The same applies to random reads. For large sequential reads and writes the case may not be quite so clear. It has been observed, however, that sequential I/O performance can be improved. An application generating many I/O requests can swamp a single disk controller, for example. If the disks in the striped volume all have their own dedicated controller, however, swamping is far less likely to occur and so performance is improved.

RAID can be implemented using either a software or a hardware RAID manager device. There are advantages and disadvantages in using either method:

- Hardware RAID generally provides higher performance as it is implemented in hardware and hence incurs less processing overhead than software RAID. Furthermore, hardware RAID is dissociated from the host system, leaving host resources free to execute applications.
- Hardware RAID is generally more expensive than software RAID.
- Software RAID can be more flexible than hardware RAID. For example, a hardware RAID manager is usually associated with a single array of disks or with a prescribed set of arrays, whereas software RAID can encapsulate any number of arrays of disks, or, if desired, only certain disks within an array.

The following sections discuss RAID configurations, known as levels. The most common RAID levels, 0, 1, 1+0 and 5 are covered in some detail, whereas less common levels are merely compared and contrasted.

RAID 0, Striped Volume

Striping spreads data across multiple physical disks. The logical disk, or volume, is divided into chunks or stripes and then distributed in a round-robin fashion on physical disks. A stripe is always one or more disk blocks in size, with all stripes having the same size.

The name RAID 0 is a contradiction in that it provides no redundancy. Any disk failure in a RAID 0 stripe causes the entire logical volume to be lost. RAID 0 is, however, the least expensive of all RAID levels as all disks are dedicated to data.

RAID 1, Mirrored Volume

The purpose of mirroring is to provide redundancy. If one of the disks in the mirror fails then the data remains available and processing may continue. The trade off is that each physical disk is mirrored, meaning that half the physical disk space is devoted to mirroring.

RAID 1+0

Also known as RAID 10, RAID 1+0 provides the highest levels of performance and resiliency. Consequently, it is the most expensive level of RAID to implement. Data continues to remain available after up to three disk failures as long as all of the disks that fail form different mirrors. RAID 1+0 is implemented as a striped array where segments are RAID 1.

RAID 0+1

RAID 0+1 is slightly less resilient than RAID 1+0. A stripe is created and then mirrored. If one or more disks fails on the same side of the mirror, then the data remains available. If a disk then fails on the other side of the mirror, however, the logical volume is lost. This subtle difference with RAID 1+0 means disks on either side can fail simultaneously yet data remains available. RAID 0+1 is implemented as a mirrored array where segments are RAID 0.

RAID 5

RAID 5 is not as resilient as mirroring yet nevertheless provides redundancy in that data remains available after a single disk failure. RAID 5 implements redundancy using a parity stripe created by performing logical exclusive or on bytes of corresponding stripes on other disks. When one disk fails, data for that disk is recalculated using the data and parity in the corresponding stripes on the remaining disks. Performance suffers however when such corrective calculations must be performed.

During normal operation, RAID 5 usually offers lower performance than RAID 0, 1+0 and 0+1, as a RAID 5 volume must do four physical I/O operations for every logical write. The old data and parity are read, two exclusive or operations are performed, and the new data and parity are written. Read operations do not suffer the same penalty and thus provide only slightly lower performance than a standard stripe using an equivalent number of disks. That is, the RAID 5 volume has effectively one less disk in its stripe because the space is devoted to parity. This means a RAID 5 volume is generally cheaper than RAID 1+0 and 0+1, because RAID 5 devotes more of the available disk space to data.

Given the performance issues, RAID 5 is not generally recommended unless the data is read-only or unless there are very few writes to the volume. Disk arrays with write caches and fast exclusive or logic engines can mitigate these performance issues however, making RAID 5 a cheaper, viable alternative to mirroring for some deployments.

RAID Levels 2, 3, and 4

RAID levels 2 and 3 are good for large sequential transfers of data such as video streaming. Both levels can process only one I/O operation at time, making them inappropriate for applications demanding random access. RAID 2 is implemented using Hamming error correction coding (ECC). This means three physical disk drives are required to store ECC data, making it more expensive than RAID 5, but less expensive than RAID 1+0 as long as there are more than three disks in the stripe. RAID 3 uses a bitwise parity method to achieve redundancy. Parity is not distributed as per RAID 5, but is instead written to a single dedicated disk.

Unlike RAID levels 2 and 3, RAID 4 uses an independent access technique where multiple disk drives are accessed simultaneously. It uses parity in a manner similar to RAID 5, except parity is written to a single disk. The parity disk can therefore become a bottleneck as it is accessed for every write, effectively serializing multiple writes.

Software Volume Managers

Volume managers such as Solaris™ Volume Manager may also be used for Directory Server disk management. Solaris Volume Manager compares favorably with other software volume managers for deployment in production environments.

Monitoring I/O and Disk Use

Disks should not be saturated under normal operating circumstances. You may use utilities such as `iostat`(1M) on Solaris and other systems to isolate potential I/O bottlenecks. Refer to Windows help for details on handling I/O bottlenecks on Windows systems.

Sizing for Multiprocessor Systems

Directory Server software is optimized to scale across multiple processors. In general, adding processors may increase overall search, index maintenance, and replication performance.

In specific directory deployments, however, you may reach a point of diminishing returns where adding more processors does not impact performance significantly. When handling extremely demanding performance requirements for searching, indexing, and replication, consider load balancing and directory proxy technologies as part of the solution.

Sizing Network Capacity

Directory Server is a network intensive application. To improve network availability for a Directory Server instance, equip the system with two or more network interfaces. Directory Server can support such a hardware configuration, listening on multiple network interfaces within the same process.

If you intend to cluster directory servers on the same network for load balancing purposes, ensure the network infrastructure can support the additional load generated. If you intend to support high update rates for replication in a wide area network environment, ensure through empirical testing that the network quality and bandwidth meet your requirements for replication throughput.

Sizing for SSL

By default, support for the Secure Sockets Layer (SSL) protocol is implemented in software. Using the software-based SSL implementation may have significant negative impact on Directory Server performance. Running the directory in SSL mode may require the deployment of several directory replicas to meet overall performance requirements.

Although hardware accelerator cards cannot eliminate the impact of using SSL, they can improve performance significantly compared with software-based implementation. Directory Server 5.2 supports the use of SSL hardware accelerators such as supported Sun Crypto Accelerator hardware.

Using a Sun Crypto Accelerator board can be useful when SSL key calculation is a bottleneck. Such hardware may not improve performance when SSL key calculation is not a bottleneck, however, as it specifically accelerates key calculations during the SSL handshake to negotiate the connection, but not encryption and decryption of messages thereafter. Refer to Appendix A, “Using the Sun Crypto Accelerator Board,” in the *Directory Server Administration Guide*, for instructions on using such hardware with a Directory Server instance.

Glossary

Refer to the *Java Enterprise System Glossary* (http://docs.sun.com/doc/816-6873) for a complete list of terms that are used in this documentation set.

Index