Deployment Guide

Sun™ ONE Directory Server

Version 5.2

816-6700-10
June 2003

Contents

Purpose of This Guide 9
Prerequisites 9
Typographical Conventions 10
Default Paths and Filenames 10
Downloading Directory Server Tools 12
Suggested Reading 12

Part 1 Directory Server Design 15

Chapter 1 Directory Server Design and Deployment Overview 17
Directory Design Overview 17
Design Process Outline 18
Directory Deployment Overview 19

Chapter 2 Planning and Accessing Directory Data 21
Introduction to Directory Data 21
What Your Directory Might Include 22
What Your Directory Should Not Include 23
Defining Your Directory Needs 23
Accessing Your Directory Data with DSML over HTTP/SOAP 24
DSMLv2 Over HTTP/SOAP Deployment 25
Performing a Site Survey 27
Identifying the Applications That Use Your Directory 28
Identify How Applications Will Access Your Directory 30
Identifying Data Sources 30
Characterizing Your Directory Data 31
Determining Directory Availability Requirements 32
Considering a Data Master Server 32
Determining Data Ownership 34

Determining Data Access 35
Documenting Your Site Survey 37
Repeating the Site Survey 38

Chapter 3 Designing the Schema 39
Sun ONE Directory Server Schema 40
Schema Design Process Overview 41
Mapping Your Data to the Default Schema 41
Viewing the Default Directory Schema 41
Matching Data to Schema Elements 42
Customizing the Schema 43
When to Extend Your Schema 44
Getting and Assigning Object Identifiers 45
Naming Attributes and Object Classes 45
Strategies for Defining New Object Classes 46
Strategies for Defining New Attributes 47
Deleting Schema Elements 48
Creating Custom Schema Files - Best Practices and Pitfalls 49
Maintaining Data Consistency 52
Schema Checking 52
Selecting Consistent Data Formats 54
Maintaining Consistency in Replicated Schema 54
Other Schema Resources 55

Chapter 4 Designing the Directory Tree 57
Introduction to the Directory Tree 57
Designing Your Directory Tree 58
Choosing a Suffix 58
Creating Your Directory Tree Structure 60
Naming Entries 67
Grouping Directory Entries and Managing Attributes 70
Static and Dynamic Groups 71
Managed, Filtered, and Nested Roles 72
Role Enumeration and Role Membership Enumeration 74
Role Scope 74
Role Limitations 76
Deciding Between Groups and Roles 77
Managing Attributes with Class of Service (CoS) 79
About CoS 80
Cos Definition Entries and CoS Template Entries 81
CoS Priorities 83
Pointer CoS, Indirect CoS, and Classic CoS 83

CoS Limitations 88
Directory Tree Design Examples 89
Directory Tree for an International Enterprise 89
Directory Tree for an ISP 90
Other Directory Tree Resources 91

Chapter 5 Designing the Directory Topology 93
Topology Overview 93
Distributing Your Data 94
Using Multiple Databases 94
About Suffixes 96
About Referrals and Chaining 100
Using Referrals 100
Using Chaining 108
Deciding Between Referrals and Chaining 110

Chapter 6 Designing the Replication Process 115
Introduction to Replication 115
Replication Concepts 116
Common Replication Scenarios 125
Single-Master Replication 125
Multi-Master Replication 127
Cascading Replication 134
Mixed Environments 138
Fractional Replication 139
Defining a Replication Strategy 142
Replication Backward Compatibility 143
Replication Survey 144
Replication Resource Requirements 145
Using Replication for High Availability 146
Using Replication for Local Availability 147
Using Replication for Load Balancing 148
Example Replication Strategy for a Small Site 153
Example Replication Strategy for a Large Site 154
Replication Strategy for a Large, International Enterprise 154
Using Replication with Other Directory Features 155
Replication and Access Control 155
Replication and Directory Server Plug-Ins 155
Replication and Chained Suffixes 157
Schema Replication 157
Replication and Multiple Password Policies 159
Replication Monitoring 159

Chapter 7 Designing a Secure Directory 163
About Security Threats 164
Unauthorized Access 164
Unauthorized Tampering 165
Denial of Service 165
Analyzing Your Security Needs 165
Determining Access Rights 166
Ensuring Data Privacy and Integrity 167
Conducting Regular Audits 167
Example Security Needs Analysis 167
Overview of Security Methods 168
Selecting Appropriate Authentication Methods 169
Anonymous Access 170
Simple Password 171
Proxy Authorization 172
Simple Password Over a Secure Connection 172
Certificate-Based Client Authentication 173
SASL-Based Client Authentication 174
Preventing Authentication by Account Inactivation 174
Designing your Password Policies 175
Password Policy Features 176
Configuring Your Password Policies 179
Designing an Account Lockout Policy 187
Designing Password Policies in a Replicated Environment 187
Designing Access Control 189
About the ACI Format 190
Default ACIs 194
Deciding How to Set Permissions 195
Requesting Effective Rights Information 198
Tips on Using ACIs 206
ACI Limitations 208
Securing Connections With SSL 209
Encrypting Attributes 211
What is Attribute Encryption? 211
Attribute Encryption Implementation 213
Attribute Encryption and Performance 214
Attribute Encryption Usage Considerations 215
Grouping Entries Securely 216
Using Roles Securely 217
Using CoS Securely 217
Securing Configuration Information 219
Other Security Resources 220

Chapter 8 Monitoring Your Directory 221
Defining a Monitoring and Event Management Strategy 222
Directory Server Monitoring Tools 222
Directory Server Monitoring 224
Monitoring Directory Server Activity 224
Monitoring Database Activity 226
Monitoring Disk Status 227
Monitoring Replication Activity 228
Monitoring Indexing Efficiency 229
Monitoring Security 230
SNMP Monitoring 231
About SNMP 231
SNMP Monitoring in Sun ONE Directory Server 233

Part 2 Directory Server Deployment Scenario and Reference Architectures 237

Chapter 9 Banking Deployment Scenario 239
Business Challenge 239
Deployment Context and Replication Topology 240
Deployment Context 240
Replication Topology 241
Performance Requirements 244
User Demands 244
Hardware Guidelines 245
Schema, Data, and Directory Information Tree Design 246
Schema 246
Data 251
Directory Information Tree 253
Security Considerations 257
Implementation 259

Chapter 10 Architectural Strategies 263
Addressing Failure and Recovery 264
Planning a Backup Strategy 265
Choosing a Backup Method 265
Choosing a Restoration Method 270
Sample Replication Topologies 273
Single Data Center 273
Two Data Centers 279
Three Data Centers 282
Five Data Centers 286

Single Data Center Using the Retro Change Log Plug-In 290

Appendix A Accessing Data Using DSMLv2 Over HTTP/SOAP 293
An Empty Anonymous DSML “Ping” Request 293
A DSML Request Issuing a User Binding 299
A DSML Search Request 300

About This Guide

Sun™ ONE Directory Server 5.2 is a powerful and scalable distributed directory server based on the industry-standard Lightweight Directory Access Protocol (LDAP). Sun ONE Directory Server software is part of the Sun Open Net Environment (Sun ONE), Sun's standards-based software vision, architecture, platform, and expertise for building and deploying Services On Demand.

Sun ONE Directory Server is the cornerstone for building a centralized and distributed data repository that can be used in your intranet, over your extranet with your trading partners, or over the public Internet to reach your customers.

Purpose of This Guide

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

This guide is divided into two parts. The first part introduces directory design concepts, including schema design, the directory tree, topology, replication, security, and monitoring. The second part presents a Sun ONE Directory Server 5.2 deployment scenario and reference replication architectures, which provide you with an insight into some of the major concerns and issues you must address.

Prerequisites

Before reading this guide we strongly recommend that you read the online release notes to obtain the latest information about new features and enhancements in this release of Sun ONE Directory Server. The release notes can be found at

```
http://docs.sun.com/doc/816-6703-10/
```

This guide assumes that you are already familiar with basic directory service and LDAP concepts and that you have read the introduction to Sun ONE Directory Server material, all of which are presented in the *Sun ONE Directory Server Getting Started Guide*.

Typographical Conventions

This section explains the typographical conventions used in this book.

`Monospaced font` - This typeface is used for literal text, such as the names of attributes and object classes when they appear in text. It is also used for URLs, filenames, and examples.

Italic font - This typeface is used for emphasis, for new terms, and for text that you must substitute for actual values, such as placeholders in path names.

The greater-than symbol (>) is used as a separator when naming an item in a menu or sub-menu. For example, Object > New > User means that you should select the User item in the New sub-menu of the Object menu.

NOTE Notes, Cautions, and Tips highlight important conditions or limitations. Be sure to read this information before continuing.

Default Paths and Filenames

All path and filename examples in the Sun ONE Directory Server product documentation are one of the following two forms:

- *ServerRoot*/... - The *ServerRoot* is the location of the Sun ONE Directory Server product. This path contains the shared binary files of Directory Server, Sun ONE Administration Server, and command line tools.

  The actual *ServerRoot* path depends on your platform, your installation, and your configuration. The default path depends on the product platform and packaging as shown in Table 1.

- *ServerRoot*/`slapd-`*serverID*`/...` - The serverID is the name of the Directory Server instance that you defined during installation or configuration. This path contains database and configuration files that are specific to the given instance.

NOTE Paths specified in this manual use the forward slash format of UNIX and commands are specified without file extensions. If you are using a Windows version of Sun ONE Directory Server, use the equivalent backslash format. Executable files on Windows systems generally have the same names with the `.exe` or `.bat` extension.

Table 1 Default *ServerRoot* Paths

Product Installation	*ServerRoot* Path
Solaris 9[1]	`/var/mps/serverroot` - After configuration, this directory contains links to the following locations: • `/etc/ds/v5.2` (static configuration files) • `/usr/admserv/mps/admin` (Sun ONE Administration Server binaries) • `/usr/admserv/mps/console` (Server Console binaries) • `/usr/ds/v5.2` (Directory Server binaries)
Compressed Archive Installation on Solaris and Other Unix Systems	`/var/Sun/mps`
Zip Installation on Windows Systems	`C:\Program Files\Sun\MPS`

1. If you are working on the Solaris Operating Environment and are unsure which version of the Sun ONE Directory Server software is installed, check for the existence a key package such as `SUNWdsvu` using the `pkginfo` command. For example: `pkginfo | grep SUNWdsvu`.

Directory Server instances are located under *ServerRoot*`/slapd-`*serverID*`/`, where *serverID* represents the server identifier given to the instance on creation. For example, if you gave the name `dirserv` to your Directory Server, then the actual path would appear as shown in Table 2. If you have created a Directory Server instance in a different location, adapt the path accordingly.

Table 2 Default Example `dirserv` Instance Locations

Product Installation	Instance Location
Solaris 9	`/var/mps/serverroot/slapd-dirserv`

Table 2 Default Example `dirserv` Instance Locations *(Continued)*

Product Installation	Instance Location
Compressed Archive Installation on Solaris and Other Unix Systems	`/usr/Sun/mps/slapd-dirserv`
Zip Installation on Windows Systems	`C:\Program Files\Sun\MPS\slapd-dirserv`

Downloading Directory Server Tools

Some supported platforms provide native tools for accessing Directory Server. More tools for testing and maintaining LDAP directory servers, download the Sun ONE 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 *Sun ONE Directory Server Resource Kit Tools Reference*.

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

Additionally, Java Naming and Directory Interface (JNDI) technology supports accessing the 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/
```

Suggested Reading

Sun ONE Directory Server product documentation includes the following documents delivered in both HTML and PDF:

- *Sun ONE Directory Server Getting Started Guide* - Provides a quick look at many key features of Directory Server 5.2.

- the *Sun ONE Directory Server Deployment Guide* - Explains how to plan directory topology, data structure, security, and monitoring, and discusses example deployments.
- *Sun ONE Directory Server Installation and Tuning Guide* - Covers installation and upgrade procedures, and provides tips for optimizing Directory Server performance.
- *Sun ONE Directory Server Administration Guide* - Gives the procedures for using the console and command-line to manage your directory contents and configure every feature of Directory Server.
- *Sun ONE Directory Server Reference Manual* - Details the Directory Server configuration parameters, commands, files, error messages, and schema.
- *Sun ONE Directory Server Plug-In API Programming Guide* - Demonstrates how to develop Directory Server plug-ins.
- *Sun ONE Directory Server Plug-In API Reference* - Details the data structures and functions of the Directory Server plug-in API.
- *Managing Servers with Sun ONE Console* - Discusses how to manage servers using the Sun ONE Administration Server and Java based console.
- *Sun ONE Directory Server Resource Kit Tools Reference* - Covers installation and features of the Sun ONE Directory Server Resource Kit, including many useful tools.

Other useful information can be found on the following Web sites:

- Product documentation online: `http://docs.sun.com/coll/S1_DirectoryServer_52`
- Sun software: `http://wwws.sun.com/software/`
- Sun ONE Services: `http://www.sun.com/service/sunps/sunone/`
- Sun Support Services: `http://www.sun.com/service/support/`
- Sun ONE for Developers: `http://sunonedev.sun.com/`
- Training: `http://suned.sun.com/`

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

Part 1

Directory Server Design

This part of the guide introduces directory design concepts and walks you through the entire design process, including data design and access, schema design, the directory tree, topology, replication, security, and monitoring. The aim of this part of the guide is to familiarize you with the design concepts in substantial detail, and to provide you with an understanding of what you must consider when designing your directory deployment.

Chapter 1

Directory Server Design and Deployment Overview

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

An introduction to basic LDAP and directory concepts and to Sun ONE Directory Server is provided in the *Sun ONE Directory Server Getting Started Guide*. This chapter provides you with an overview of the directory design and deployment process, and is divided into the following sections:

- Directory Design Overview
- Directory Deployment Overview

Directory Design Overview

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

The flexibility of Sun ONE Directory Server allows you to rework your design to meet unexpected or changing requirements, even after you deploy Directory Server. That said , the more modifications you can avoid through good design, the better.

Design Process Outline

The design process is broken into six steps:

- Planning and Accessing Directory Data

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

- Designing the Schema

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

- Designing the Directory Tree

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

- Designing the Directory Topology

  Topology design involves determining how you divide your directory tree among multiple physical Directory Servers and how these servers communicate with one another. Chapter 5, “Designing the Directory Topology,” describes the general principles behind topology design, discusses using multiple databases, describes the mechanisms available for linking your distributed data together, and explains how Directory Server itself 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, “Designing the Replication Process”, describes how replication works, what kinds of data you can replicate, common replication scenarios, and tips for building a highly available directory service.

- Designing a Secure Directory

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

- Monitoring Your Directory

  Up to this point, you have concentrated on designing a directory service that addresses your requirements and is as secure as possible. However, if you cannot monitor your directory service satisfactorily, then you will not be able to either evaluate the success of your directory service deployment or follow the day-to-day directory activities. Chapter 8, “Monitoring Your Directory” discusses how to monitor your directory using SNMP, the 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 your 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 *Sun ONE 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. Sun ONE Directory Server 5.2 allows you to access the directory data either via LDAP or DSML, which opens up the possibilities in terms of the applications that can interact directly with your directory data.

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

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

Introduction to Directory Data

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

- It is read more often than written.

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

- It is expressible in attribute-data format (for example, `surname=jensen`).

- It is of interest to more than one audience.

  For example, an employee's name or the physical location of a printer can be of interest to many people and applications.

- It will be accessed from more than one physical location.

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

What Your Directory Might Include

Examples of data you can put in your directory are:

- Contact information, such as telephone numbers, physical addresses, and 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 perfectly 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 the directory works best for read operations, you should avoid placing rapidly changing information in the directory. Reducing the number of write operations occurring in your directory improves overall search performance.

Defining Your Directory Needs

When you design your directory data, 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 influences how you structure and distribute the data in your directory.

As you plan, consider these points:

- What do you want to put in your directory today? What immediate problem do you hope to solve by deploying a directory? What are the immediate needs of the directory-enabled application you use?
- What do you want to put in your directory in the near future? For example, your enterprise might use an accounting package that does not currently support LDAP, but that you know will be LDAP-enabled 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 from your current customers. Maybe future customers will want to use your directory to store JPEG images. While this is the most difficult case of all to consider, doing so may pay off in unexpected ways. At a minimum, this kind of planning helps you identify data sources you might otherwise not have considered.

Accessing Your Directory Data with DSML over HTTP/SOAP

In contrast to previous versions of Directory Server that only allowed you to access your directory data using the Lightweight Directory Access Protocol (LDAP), Sun ONE Directory Server 5.2 also allows you to access your directory data using Directory Service Markup Language version 2 (DSMLv2) over HTTP/SOAP.

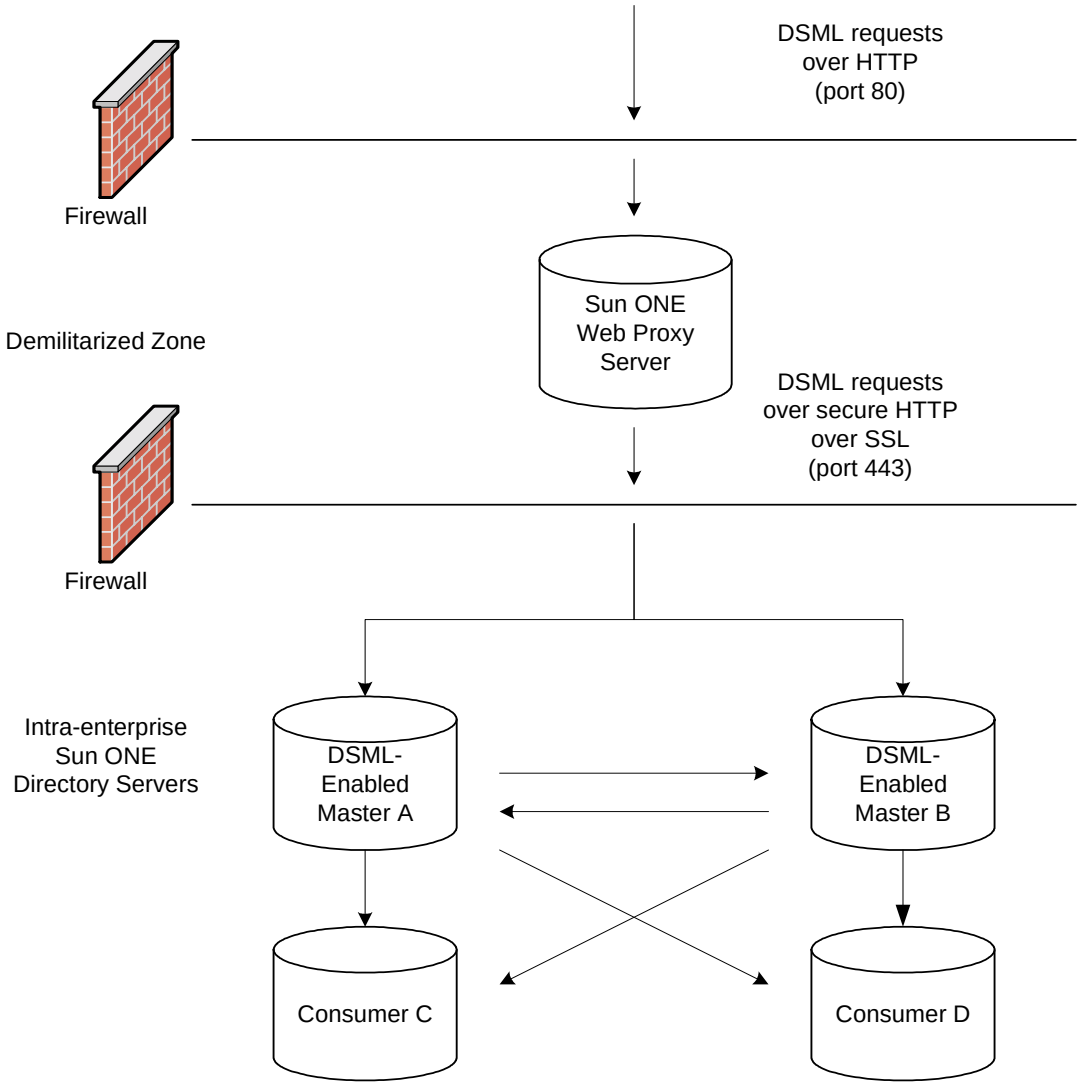
DSMLv2 is a markup language, that is, a vocabulary and schema that enables users to describe the structure and content of directory services data operations in an eXtensible Markup Language (XML) document. DSMLv2 standardizes the way directory services information is represented in XML, and with such a recognized standard, applications can be written to make use of DSMLv2 and capture the scalability, replication, security and management strengths of directory services. Given that DSMLv2 is *not* an access protocol, DSMLv2 still relies on an access protocol to actually access the data contained in the directory. 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.

The fact that it is now possible to access your directory using DSMLv2 over HTTP/SOAP, changes somewhat the scope of possible applications with which your directory can interact, as they will not have to be LDAP applications. The following section presents the new DSMLv2 over HTTP/SOAP access possibilities in more detail. For examples on how to access and search your data using DSMLv2 over HTTP/SOAP, see Appendix A, “Accessing Data Using DSMLv2 Over HTTP/SOAP”.

DSMLv2 Over HTTP/SOAP Deployment

One possible deployment using DSML-enabled Directory Servers and Sun ONE Web Proxy Server, which would allow non-LDAP clients to interact with your directory data, is presented in Figure 2-1 on page 25:

Figure 2-1 Sample DSML-Enabled Directory Server Deployment



Update requests in DSML arriving from non-LDAP client applications first cross a firewall over HTTP port 80 and enter into a demilitarized zone. From there Sun ONE Web 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.

The idea behind this deployment is primarily to allow non-LDAP applications to interact with your directory data by performing directory operations. If the requests from your clients are solely lookup requests, then it is of no importance as to whether your DSML-enabled Directory Servers hold read-only or read-write copies of your data, because both would be able to process the lookup requests. However, if your non-LDAP client issues modification requests, then it is important for the DSML-enabled Directory Servers to hold read-writе (master replica) copies of the data. This is because, in the context of replication, the default behavior for a consumer (which holds a read-only copy of the data) receiving a modification request, is to return a referral with a list of LDAP URLs for the possible masters that could satisfy the client's modification request. Returning an LDAP URL over HTTP to a non-LDAP client application does not make sense and would not really fulfill the initial objective of keeping client/directory traffic LDAP-free, which is why read-write copies are preferable. The deployment depicted in Figure 2-1 on page 25, holds read-write copies of the data on the DSML-enabled Directory Servers Master A and Master B that process your client's modification requests and then replicate the data to the non-DSML enabled Consumers C and D.

As previously stated, DSML access over HTTP/SOAP opens up your directory data to the worlds of XML and web services, but it may do so with increased security risks. The DSML front end of Directory Server constitutes a restricted HTTP server, because 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, we recommend you take into account the following security considerations when including DSML-enabled Directory Servers in your deployment:

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

Performing a Site Survey

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

- Identify the applications that use your directory.

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

- Identify how the applications will access your 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 your directory needs to contain.

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

- Determine the level of service you need to provide.

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

  For more information about replication, refer to Chapter 6, “Designing the Replication Process” on page 115. For more information on chaining, refer to Chapter 5, “Designing the Directory Topology.”

- Identify a data master.

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

- Determine data ownership.

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

- Determine data access.

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

- Document your site survey.

  Because of the number of organizations that can be affected by the directory, it may be helpful to create a directory deployment team that includes representatives from each affected organization. This team performs the site survey.

  Corporations generally have a human resources department, an accounting 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 your site 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 (comprising representatives from each location).

Identifying the Applications That Use Your Directory

Generally, the applications that access your directory and the data needs of these applications drive the planning of your directory contents. 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 to be available in the directory. 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, and job title.
- 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.

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

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

- The data required by your various legacy applications and your user population.

- The ability of your legacy applications to communicate with an LDAP directory.

Identify How Applications Will Access Your Directory

If non-LDAP applications must be able to interact with your directory data by performing directory operations, consider accessing your directory using DSML over HTTP/SOAP. However, if your client applications are LDAP applications, LDAP access will be your choice. The mode of access you choose will depend on which applications use your directory.

Identifying Data Sources

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

- Identify organizations that provide information.

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

- Identify the tools and processes that are information sources.

  Some common sources for information are networking operating systems (Windows, 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 Your Directory Data

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

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

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

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

Table 2-3 Directory Data Characteristics

Data	Format	Size	Owner	Related to
Employee Name	Text string	128 characters	Human resources	User’s entry
Fax number	Phone number	14 digits	Facilities	User’s entry

Table 2-3 Directory Data Characteristics

Data	Format	Size	Owner	Related to
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 upon the expectations of the people who rely on directory-enabled applications. To determine the level of service each application expects, first determine how and when the application is used.

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

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

Considering a Data Master Server

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

Data Mastering for Replication

Sun ONE Directory Server allows you to contain master sources of information on more than one server. If you use replication, decide which server is the master source of a piece of data. Sun ONE Directory Server supports multi-master configurations, in which more than one server can be a master source for the same piece of data. For more information about replication and multi-master replication, see "Designing the Replication Process," on page 115.

In the simplest case, put a master source of all of your data on two Directory Servers and then replicate that data to one or more consumer servers. Having two master servers provides safe failover in the event that a server goes off-line. In more complex cases, you may want to store the data in multiple databases, so that the entries are mastered by a server close to the applications 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 ONE 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 ONE sales representative for more information about Sun ONE 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 the how you maintain data masters, keep it simple and consistent. For example, you should not attempt to master data in multiple sites, then automatically exchange data between competing applications. Doing so leads to a “last change wins” scenario and increases your administrative overhead.

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

Determining Data Ownership

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

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

  This subset of information might include their passwords, descriptive information about themselves and their role within the organization, their automobile license plate number, and contact information such as telephone numbers or office numbers.
- Allow a person’s manager to write to some strategic subset of that person’s information, such as contact information or job title.
- Allow an organization’s administrator to create and manage entries for that organization.

  This approach makes your organization’s administrators your directory content managers.

- Create roles that give groups of people read or write access privileges.

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

  For more information about roles and grouping entries, refer to “Designing the Directory Tree,” on page 57.

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

For information on setting access control for your directory, see Chapter 7, “Designing a Secure Directory,” on page 163.

Determining Data Access

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

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

- Can the data be read anonymously?

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

- Can the data be read widely across your enterprise?

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

For more information about access control, refer to "Designing Access Control," on page 189.

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

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

  For information about groups and roles, see Chapter 4, "Designing the Directory Tree," on page 57. For information about access controls, see Chapter 7, "Designing a Secure Directory," on page 163.

As you make these decisions for each piece of directory data, you define a security policy for your directory. Your decisions depend upon the nature of your site and the kinds of security already available at your site. For example, if your site has a firewall or no direct access to the Internet, you may feel freer to support anonymous access than if you are placing your directory directly on the Internet.

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

The creation of a security policy and the way you implement it is described in detail in Chapter 7, "Designing a Secure Directory," on page 163.

Documenting Your Site Survey

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

A basic data tracking example is provided in Table 2-4 on page 37. 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

Looking at the row representing the employee name data, we see 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 human resources group can change, add, and delete employee names in the directory.

- IS Writable

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

Repeating the Site Survey

Finally, you may need to run more than one site survey, particularly if your enterprise has offices in multiple cities or countries. You may find your informational needs to be so complex that you have to allow several different organizations to keep information at their local offices rather than at a single, centralized site. In this case, each office that keeps a master copy of information should run its own site survey. After the site survey process has been completed, the results of each survey should be returned to a central team (probably consisting of representatives from each office) for use in the design of the enterprise-wide data schema model and directory tree.

Chapter 3

Designing the 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 you can store in your 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 the integrity of the data chained suffix you store in your directory.

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

- Sun ONE Directory Server Schema
- Schema Design Process Overview
- 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 *Sun ONE Directory Server Reference Manual*. For information on replicating schema between servers, refer to “Schema Replication,” on page 157.

Sun ONE Directory Server Schema

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

The predefined schema included with Directory Server contains 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 needs, you may need to extend it with new object classes and attributes to accommodate the unique needs of your directory. Refer to “Customizing the Schema,” on page 43 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 programmatically retrieve the schema and adapt their behavior based on it. The global set of schema for Directory Server can be found in the entry named `cn=schema`.

The Directory Server schema 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 ONE for the 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 (objectlass $ sn $ cn) MAY (description
 $ seealso $ telephoneNumber $ userPassword) X-ORIGIN 'RFC 2252' )
```

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

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

Schema Design Process Overview

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

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

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

Mapping Your Data to the Default Schema

The data you identified during your site survey, as described in “Performing a Site Survey,” on page 27, must be mapped to the existing directory default schema. This section describes how to view the existing default schema and provides a method for mapping your data to the appropriate existing schema elements.

If you find elements in your schema that do not match the existing default schema, you may need to create custom object classes and attributes. Refer to “Customizing the Schema,” on page 43 for more information.

Viewing the Default Directory Schema

The schema provided with Sun ONE Directory Server 5.2 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 ONE products. The LDAPv3 standard user and organization schema can be found in the `00core.ldif` file. The configuration schema used by earlier versions of the directory can be found in the `50ns-directory.ldif` file.

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

It is also important to realize that any changes made manually will *not* be replicated until other changes are made using either LDAP or the Directory Server console.

Matching Data to Schema Elements

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

- Identify the type of object the data describes.

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

- Select a similar object class from the default schema.

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

- Select a similar attribute from the matching object class.

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

- Identify the unmatched data from your site survey.

  If there are some pieces of data that do not match the object classes and attributes defined by the default directory schema, you will need to customize the schema. See "Customizing the Schema," on page 43 for more information.

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

Table 3-1 Data Mapped to Default Directory Schema

Data	Owner	Object Class	Attribute
Employee name	HR	person	cn(commonName)

Table 3-1 Data Mapped to Default Directory Schema

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

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

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

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

Customizing the Schema

You can extend the standard schema if it proves to be too limited for your directory needs. The Directory Server Console can help you manage the schema definition. For more information, refer to Chapter 9, "Extending the Directory Schema" in the *Sun ONE Directory Server Administration Guide*.

Keep the following rules in mind when customizing your schema:

- Reuse existing schema elements whenever possible. For a complete list of the existing schema elements, refer to the "Directory Server Schema" in the *Sun ONE Directory Server Reference Manual*.
- 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.

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

Getting and Assigning Object Identifiers

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

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

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

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

  An OID registry is a list you maintain that gives the OIDs and descriptions of the OIDs used in your directory schema. This ensures that no OID is ever 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 your schema elements and existing schema elements by including a unique prefix on all of your 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

There are two ways you can create new object classes:

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

You may find it easiest to mix the two methods.

For example, suppose your site wants to create the attributes `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 ONE OID prefix and must not be used in the deployed product. To create your own new object classes, you must get your own OID. For more information, refer to “Getting and Assigning Object Identifiers,” on page 45.

Choose the strategy for defining new object classes that works for you. 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 a more careful and rigid data design.

  Rigid data design forces you to consider the object class structure on which every piece of data will be placed. Depending on your personal preferences, you will find this to be either helpful or cumbersome.

- Single `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 sensical 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 what object class(es) it inherits.

Strategies for Defining New Attributes

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

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

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

Deleting Schema Elements

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

If you extend the schema and find that you do not use the new elements, you are free to delete these unused elements. Before removing schema elements you must make sure that no entry in the directory uses them. The easiest way to ensure that no entries are using the schema element you want to delete 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 may 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 that use the definition 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, there are several things you need to bear in mind when creating custom schema files, especially when replication is involved which are highlighted in the following list:

- When adding new schema elements, all attributes need to be defined before they can be used in an object class. You can define attributes and object classes in the same schema file.
- Each custom attribute or object class you create should be defined in only one schema file. This prevents the server from overriding any previous definitions when it loads the most recently created schema (as the server loads the schema in numerical order first, then alphabetical order).
- 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 it causes the new elements to be written automatically to the `99user.ldif` file. As a result of this behavior any other schema definition changes you may have made in custom schema files may be overwritten, hence the recommended best practice of adding all your schema definitions to the `99user.ldif` file. This way you avoid possible duplications of schema elements and the danger of schema changes being overwritten at a later date.
- Given that Directory Server loads schema files in alpha-numerical order, that is, with numbers being loaded first, make sure that you name your custom schema files as follows:

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

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

  If you name your schema file with a number which is lower than the standard schema files, the server may encounter errors when loading the schema, and what is more, all standard attributes and object classes will be loaded only after your custom schema elements have been loaded.

- Make sure that the custom schema files you create 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, then the next time you updated the schema using LDAP or the 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 of thumb, you should identify the custom schema elements you are adding with the following two items:

  - `'user defined'` in the `X-ORIGIN` field of your custom schema files,
  - AND something which is more descriptive such as `'Example.com Corporation defined'` in the `X-ORIGIN` field, to facilitate future understanding of the custom schema element context. For example `X-ORIGIN ('user defined' 'Example.com Corporation defined')`.

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

  We recommend using a more descriptive identifier to complement `'user defined'` because it will help you identify and understand the origin of the new schema and what other schema it relates to. If you have nothing more descriptive than the `'user defined'` value which is added automatically by the server if you happen to be adding your customer schema definitions using LDAP or the Directory Server console, then you may have difficulty understanding what the schema relates to at a later date.

- It is important to propagate any custom schema files you create or modify manually to all of your servers, because these changes will not be replicated automatically.

  When you make changes to your directory schema, your 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, will push any schema changes. For custom schema files the server only maintains 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 the `99user.ldif` file will not be replicated to the other servers in your topology. For this reason you must propogate any custom schema files you create or modify to all other servers to ensure that all schema information is present throughout the topology.

To propagate any changes to custom schema files you can either:

- Replicate the changes by running the `schema_push.pl` script, which requires a restart of each server, or
- Manually copy these custom schema files to all of your servers, which requires a restart of each server.

If you choose to allow the replication process to replicate any new custom schema definitions to all of your servers, make sure that you maintain your schema on one master only. When schema definitions are replicated to a consumer server where 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 choose instead to copy your schema files to each server you must remember to copy the files EACH time changes are made. If you do not copy them each time changes are made, it is possible that the changes will be replicated and stored in the `99user.ldif` file on the consumer. Having the changes in the `99user.ldif` file may make schema management difficult, as some attributes will appear in two separate schema files on a consumer, once in the original custom schema file you copied from the supplier and again in the `99user.ldif` file after replication.

- If you do not want custom schema elements to be replicated to other servers in your replication topology, you must:
  - define the schema elements you do not want to replicate in a separate file,
  - NOT identify them 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 It is necessary to turn this `nsslapd-schema-repl-useronly` attribute to on when replicating to 5.0 or 5.1 Directory Servers.

For more information about replicating schema, see “Schema Replication,” on page 157.

Maintaining Data Consistency

Maintaining data consistency within Directory Server aids LDAP client applications in locating directory entries. For each type of information you store in the directory, you should 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 very difficult to locate information in your directory tree efficiently.

You can maintain schema consistency in the following ways:

- Use schema checking to ensure attributes and object classes conform to the schema rules.
- Select and apply a consistent data format.

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

Schema Checking

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

NOTE Schema checking 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 has an attribute called `nsslapd-valuecheck` which allows you to check only those attributes with the DN syntax. However, this attribute is turned off by default which which means that no attribute values are checked.

By default, the directory enables schema checking. We do not recommend turning it off on a server that is accepting client updates. For information on turning schema checking on and off, refer to “Turning Schema Checking On and Off” in the *Sun ONE Directory Server Administration Guide.*

With schema checking on, you must be attentive to required and allowed attributes as defined by the object classes. Object class definitions usually contain at least one required attribute, and one or more optional attributes. Optional attributes are attributes that you are allowed, but not required, to add to the directory entry. If you attempt to add an attribute to an entry that is neither required nor allowed according to the entry's object class definition, then Directory Server returns an object class violation message.

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

NOTE Bear the following items in mind when configuring the schema checking functionality:

- Generally speaking you replicate all required attributes for each entry as defined in the schema, to avoid schema violations, but should you want to filter out some of the required attributes using the fractional replication functionality, then you need to disable schema checking.
- Having schema checking enabled with fractional replication can prevent you from being able to initialize off line, that is from an ldif file, because it would not allow you to load the ldif file if required attributes were filtered out.
- Turning schema checking off may have the added benefit of improving performance.
- When 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 replicas (read-write) replicas on the same server instance.
- Since 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 and therefore, it will not correspond to the fractional replication configuration being applied.

Selecting Consistent Data Formats

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

With the LDAP protocol and Sun ONE 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. Therefore, 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 157.

Other Schema Resources

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

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

Chapter 4

Designing the Directory Tree

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 your own directory tree, and includes the following sections:

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

Introduction to the Directory Tree

The directory tree provides a way for your 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. Designing an adequate directory tree upfront will be less time-consuming than redesigning an inadequate directory tree after deployment.

A well-designed directory tree provides the following:

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

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

In addition to these considerations, you must bear in mind that your replication configuration possibilities will be restricted by the type of directory tree structure that you choose. You must carefully define your directory tree partitions for replication to work, and if you only want to replicate portions of your directory tree, that desire needs to be taken into account during your directory tree design process. In the same way, if you plan to use access controls on branch points, you need to take that into account at directory tree design time.

Designing Your Directory Tree

This section guides you through the major decisions you make during the directory tree design process. The directory tree design process involves choosing a suffix to contain your data, determining the hierarchical relationship amongst data entries, and naming the entries in your directory tree hierarchy. The following sections describe the directory tree design process in more detail:

- Choosing a Suffix
- Creating Your Directory Tree Structure
- Naming Entries

Choosing a Suffix

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

By default, the standard Sun ONE Directory Server deployment contains multiple suffixes, one for storing data and the others for data needed by internal directory operations (such as configuration information and your directory schema). For more information on these standard directory suffixes, refer to Chapter 3, “Creating Your Directory Tree” in the *Sun ONE Directory Server Administration Guide*.

Suffix Naming Conventions

All entries in your directory should be located below a common base entry that is called the suffix. You can have more than one suffix in your directory and it is important to consider the following recommendations for naming the directory suffix:

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

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

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

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

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

- `c (countryName)`

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

- `l (localityName)`

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

- `st (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 allows for 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
```

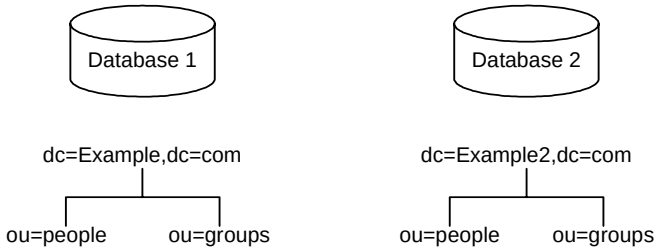
For more information on these attributes, see Table 4-1 on page 62.

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

Naming Multiple Suffixes

Each suffix that you use with your directory is a unique directory tree. You can create multiple directory trees stored in separate databases served by Directory Server. For example, you could create separate suffixes for the Example.com and the Example2.com and store them in separate databases as illustrated in Figure 4-1 on page 60:

Figure 4-1 Two Suffixes Stored in Two Different Databases



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

Creating Your Directory Tree Structure

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

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

- Branching Your Directory
- Identifying Branch Points
- Replication Considerations
- Access Control Considerations

Branching Your Directory

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

The following guidelines are specific to designing your directory tree hierarchy:

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

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

- Use functional or generic names rather than actual organizational names for your branch points.

  Names change and you do not want to have to change your directory tree every time your enterprise renames its divisions. Instead, use generic names that represent the function of the organization (for example, use `Engineering` instead of `Widget Research and Development`).

- If you have multiple organizations that perform similar functions, try creating a single branch point for that function instead of branching based along divisional lines.

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

Following are specific guidelines for the enterprise and hosting environment.

Branching in an Enterprise Environment

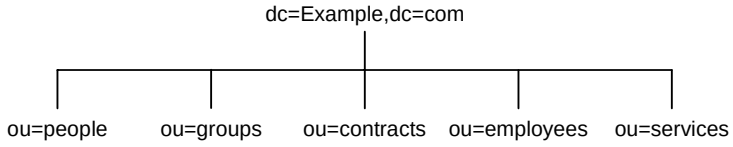
Name changes can be avoided if you base your directory tree structure on information that is not likely to change. For example, base the structure on types of objects in the tree rather than organizations. Some of the objects you might use to define your structure are:

- `ou=people`
- `ou=groups`
- `ou=contracts`

- ou=employees
- ou=services

A directory tree organized using these objects in an example enterprise Example.com, might appear as illustrated in Figure 4-2 on page 62:

Figure 4-2 Sample Directory Information Tree Using 5 Branching Points



It is wise to try to use only the traditional attributes (shown in Table 4-1 on page 62). Using traditional attributes increases the likelihood of retaining compatibility with third-party LDAP client applications. Using the traditional attributes also means that they will be known to the default directory schema, which makes it easier to build entries for the branch DN.

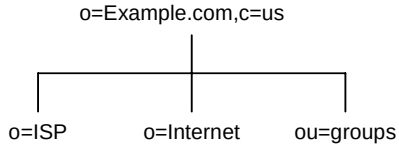
Table 4-1 Traditional DN Branch Point Attributes

Attribute Name	Definition
c	A country name.
o	An organization name. This attribute is typically used to represent a large divisional branching such as a corporate division, academic discipline (the humanities, the sciences), subsidiary, or other major branching within the enterprise. You should also use this attribute to represent a domain name as discussed in “Suffix Naming Conventions,” on page 59.
ou	An organizational unit. This attribute is typically used to represent a smaller divisional branching of your enterprise than an organization. Organizational units are generally subordinate to the preceding organization.
st	A state or province name.
l	A locality, such as a city, country, office, or facility name.
dc	A domain component as discussed in “Suffix Naming Conventions,” on page 59.

Branching in a Hosting Environment

For a hosting environment, create a tree that contains two entries of the object class `organization(o)` and one entry of the `organizationalUnit(ou)` object class beneath the suffix. For example, the ISP Example.com branches their directory as illustrated in Figure 4-3 on page 63:

Figure 4-3 ISP Example.com Directory Information Tree



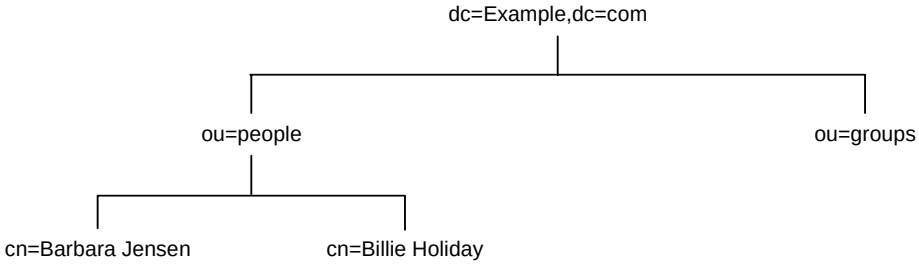
Identifying Branch Points

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

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

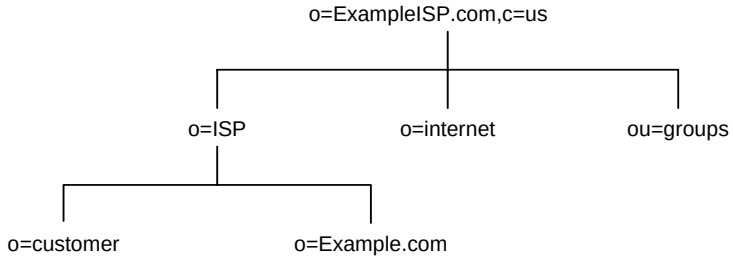
Each attribute-data pair represents a branch point in your directory tree. For example, the directory tree for the enterprise Example.com Corporation appears as illustrated in Figure 4-4 on page 63:

Figure 4-4 Example.com Corporation Directory Information Tree



Likewise the directory tree for ExampleHost.com, an internet host would appear as illustrated in Figure 4-5 on page 64:

Figure 4-5 ExampleHost.com Internet Host Directory Information Tree



Beneath the suffix entry, `o=ExampleHost.com,c=US`, the tree is split into three branches. The ISP branch contains customer data and internal information for Example.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. That is, if `l (localityName)` is subordinate to `o (organizationName)`in one part of your directory tree, then make sure `l` is subordinate to `o` in all other parts of your directory.

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

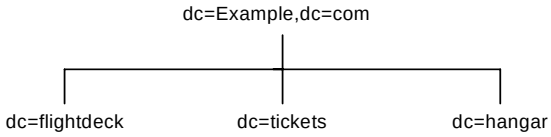
Replication Considerations

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

For example, in an enterprise environment you can organize your directory tree so that it corresponds to the network names in your enterprise. Network names tend 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.

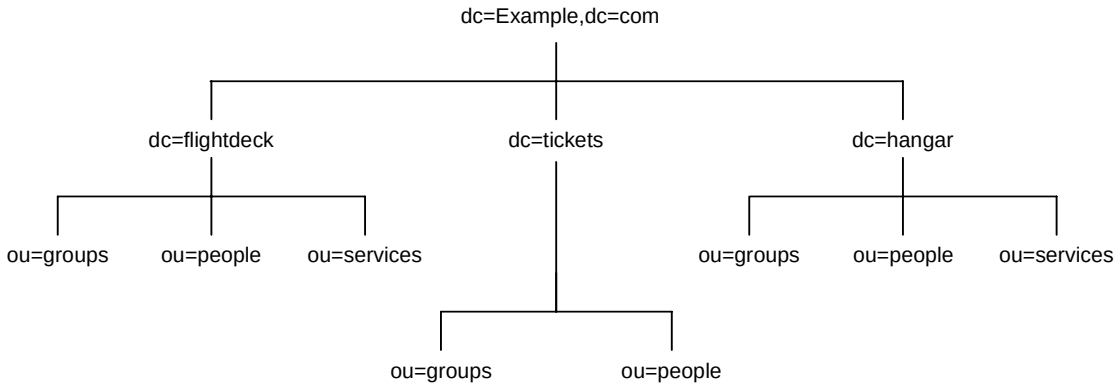
For example, Example.com Corporation has three primary networks known as `flightdeck.Example.com`, `tickets.Example.com`, and `hanger.Example.com`. They initially branch their directory tree as illustrated in Figure 4-6 on page 65:

Figure 4-6 Three Primary Networks in Example.com Corporation DIT



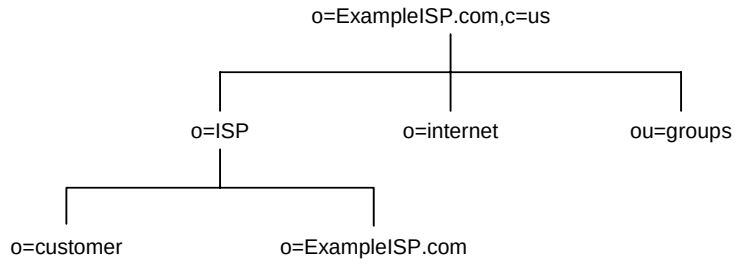
After creating the initial structure of the tree, they create additional branches as illustrated in Figure 4-7 on page 65:

Figure 4-7 Detailed View of Three Primary Networks in Example.com Corporation DIT



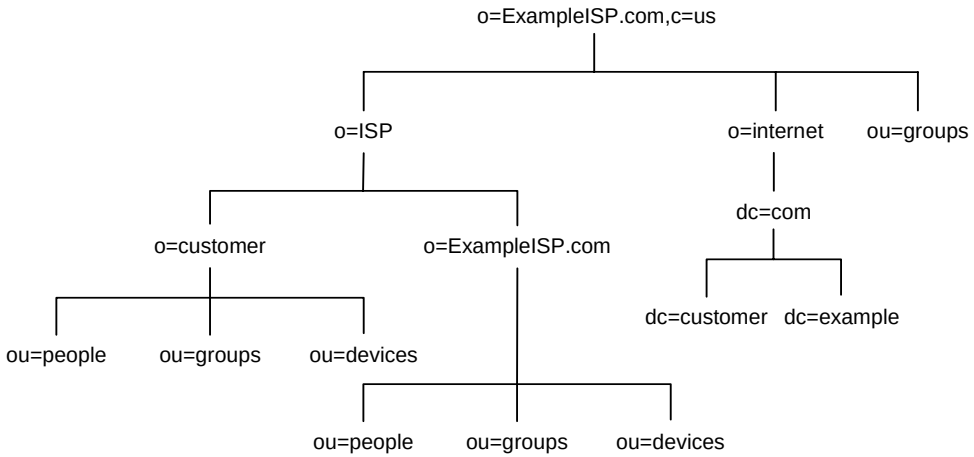
As another example, ExampleISP.com, an ISP company, branch their directory as illustrated in Figure 4-8 on page 66:

Figure 4-8 Directory Information Tree for ExampleISP.com



After creating the initial structure of their directory tree, they create additional branches as illustrated in Figure 4-9 on page 66:

Figure 4-9 Detailed View of the DIT for ExampleISP.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 be used to enable certain types of access control. As with replication, it is easier to group together similar entries and then administer them from a single branch.

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

However, ACI filters can be difficult to manage. You must decide which method of access control is best suited to your directory: organizational branching in your directory tree hierarchy, ACI filters, or a combination of the two. To facilitate the management of ACIs in your directory, Sun ONE Directory Server 5.2 provides functionality called getEffectiveRights to request which access control rights a given user has to directory entries and attributes. This getEffectiveRights functionality helps to ease user administration, access control policy verification, and debugging and is explained in more detail in Chapter 7, “Designing a Secure Directory.”

Naming Entries

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

Your entry names should adhere to the following rules:

- The attribute you select for naming should be unlikely to change.
- The name must be unique across your directory. A unique name ensures that a DN can refer to at most one entry in your directory.

When creating entries, define the RDN within the entry. By defining at least the RDN within the entry, you can locate the entry more easily. This is because searches are not performed against the actual DN but rather 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 allowing you to instantly recognize the person associated with the entry, it might not be unique in an organization where two people have identical names. This quickly 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 can lead to awkward common names for large directories and can be difficult to maintain.

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

- `uid`

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

- `mail`

  Use the `mail` attribute to contain the value for the person's 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 some unique value that you can use with the `uid` attribute. For example, you would use the `mail` attribute instead of the `uid` attribute if your enterprise does not assign employee numbers or user IDs for temporary or contract employees.

- `employeeNumber`

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

Whatever you decide to use for an attribute-data pair for person entry RDNs, you should make sure that they are unique, permanent values.

Considerations for Person Entries in a Hosted Environment

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

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

Placing Person Entries in the DIT

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

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

Naming 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, you need to include the following attributes in the organization's entry:

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

Naming Other Kinds of Entries

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

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

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

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

There does not have to be any correspondence between the attributes used for the entry's DN and the attributes actually used in the entry. However, having identifying attributes visible in the DN simplifies the administration of your directory tree.

Grouping Directory Entries and Managing Attributes

Your directory tree organizes the information of your 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 you with groups and roles that are also grouping mechanisms but that 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
- Role Limitations
- Deciding Between Groups and Roles
- Managing Attributes with Class of Service (CoS)
- About CoS
- Cos Definition Entries and CoS Template Entries
- CoS Priorities
- Pointer CoS, Indirect CoS, and Classic CoS
- CoS Limitations

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 entries are located, which makes this grouping mechanism flexible, especially when your organization is one that undergoes frequent changes. Once you know the name of a group, it is easy to retrieve all of its member entries. The following list discusses the characteristics of both static and dynamic groups and helps you to understand in which cases it might be preferable to use each type of groups:

- 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 and the `uniqueMember` attribute syntax comprises a DN followed by an optional unique identifier, against which the server would conduct its membership check. However, today Directory Server only supports the `groupOfNames` (`member` attribute) native access control processing. For more information on the syntax of the `uniqueMember` attribute refer to RFC 2256: A Summary of the X.500(96) User Schema for use with LDAPv3 http://www.ietf.org/rfc/rfc2256.txt).

- Static groups are suitable for groups with few members, such as the group of directory administrators, and as a result not for groups with thousands of members. We recommend that you avoid creating static groups with more than 20,000 members, because they will have very poor performance. For groups of this size and more, we recommend using dynamic groups or roles. If you must use static groups to define groups with 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 given group. These groups are dynamic because membership is defined every 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 either 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 By inserting the DN of another group against the `uniqueMember` attribute of a dynamic group you can place groups inside other groups, i.e., you can *nest* groups.

Although both types of groups may identify members anywhere in the directory, we recommend that the group definitions themselves 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 a new entry grouping mechanism that automatically identify all roles of which any entry is a member. Each role has *members*, or entries that possess the role. As with groups, you can specify role members either explicitly or dynamically. When you retrieve any entry in the directory, you can immediately know the roles to which it belongs, since the roles mechanism automatically generates the `nsRole` attribute containing the DN of all role definitions in which the entry is a member. This overcomes the main disadvantage of the group mechanism.

The role mechanism is very simple to use from the client perspective because the directory automatically computes all role membership. Every entry belonging to a role will be given the `nsRole` virtual attribute whose values are the DNs of all roles for which the entry is a member. The `nsRole` attribute is said to be virtual because

it is generated on-the-fly by the server and never actually 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.

Sun ONE 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 - Allow 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 given 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 that role are entries in the 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. Whenever 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

A nested role lists the definition entries of other roles and combines all the members of their roles. In other words, 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. Nested roles allow you to create roles that contain other roles.

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 any entry belongs. The `nsRole` attribute may only be used by the roles mechanism and is protected against all modifications. However, it can be read, so should your security requirements be such that you do not want to expose your role membership to read access, you may want to define access controls to protect it against reading.

Role Membership Enumeration

In contrast to previous releases of Directory Server, you can now perform searches on virtual attributes, which means that it is now possible to perform a search on the `nsRole` attribute, and enumerate the members of your role. However, it is important to bear in mind 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 for example will not be indexed and will result in poorer performance. Since the `nsRoleDN` attribute is indexed by default, searches on managed roles should be relatively efficient. However, with regard to filtered and nested roles, where filters can contain both indexed and non-indexed attributes, care should be taken to ensure that the filter contains at least one indexed attribute so as not to launch a non-indexed search.

Role Scope

In previous releases of Directory Server the scope of role was limited to the subtree of the role definition. If you wanted to have shared roles across different subtrees, they had to be at the root of the tree, which meant that cross subtree roles were limited and complicated to administer. Imagine a engineer working for the engineering department of a large enterprise Example.com, who needed to access an application controlled by the sales department. If a role had been created for the sales administrators to control access to this application within the sales subtree, then the engineer who did not belong to the sales subtree, but instead the engineering subtree, had no way of accessing the application in question, as role scope was strictly subtree limited. Given that groups did not (and still do not) have

the same scope limitations, the solution would have been to add the engineer to the group of users in the sales organization that had access to the application. However, as this solution was necessarily a group based one, the advantages of the roles mechanism would have been lost as a result.

Sun ONE Directory Server 5.2 provides a new attribute that allows the scope of a role to be extended beyond the subtree of the role definition entry. This new single-valued attribute `nsRoleScopeDN` contains the DN of the scope we want to add to an existing role.

NOTE The new scope extending `nsRoleScopeDN` attribute can only be added to a nested role.

Our previous example of the engineer working for Example.com requiring access to a sales application will help illustrate how the scope of a role can now be extended. Imagine the two main subtrees in the Example.com directory tree: `o=eng,dc=example,dc=com` for the engineering subtree and `o=sales, dc=example,dc=com` for the sales subtree. To extend the scope of a sales subtree role called `SalesAppManagedRole` that governs access to a sales application, to include an engineer user in the engineering subtree, you would need to do the following:

1. Create a role for the engineer user in the engineering subtree, for example, `EngineerManagedRole`. (In our example we have chosen to create 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` role and the initial `SalesAppManagedRole` role.
3. Add the new `nsRoleScopeDN` attribute to the new `SalesAppPlusEngNestedRole` nested role, with the DN of the engineering subtree scope you want to add, i.e. in this case `o=eng,dc=example,dc=com`.

The new `SalesAppPlusEngNestedRole` nested role would read as follows:

```
dn:cn=SalesAppPlusEngNestedRole,dc=example,dc=com
objectclass:LDAPsubentry
objectclass:nsRoleDefinition
objectclass:nsComplexRoleDefinition
objectclass:nsNestedRoleDefinition
nsRoleDN:cn=SalesAppManagedRole,o=sales,dc=example,dc=com
nsRoleDN:cn=EngineerManagedRole,o=eng,dc=example,dc=com
nsRoleScopeDN:o=eng,dc=example,dc=com
```

From an access control point of view, it is essential that the necessary permissions be granted to the engineering user wanting to access the sales application, so that access can in fact be gained to the `SalesAppPlusEngNestedRole` role, and in turn the actual sales application. Similarly, in the context of replication you must be sure to replicate the entire scope of your role, because if you fail to replicate the extended scope you will encounter problems.

NOTE By only allowing nested roles to have extended scope, an administrator who previously managed roles in one domain will only have rights to make use of the roles that already exist in the other domain, and will not therefore be able to create an arbitrary role to appear in arbitrary users in the other domain.

Role Limitations

When creating roles to support your directory service, you need to 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 further information see “About CoS,” on page 80. However, the specifier attribute in a Cos definition may reference the `nsRole` attribute generated by a role definition. For further information concerning the creation of role-based attributes, refer to “Creating Role-Based Attributes” in Chapter 5 of the *Sun ONE Directory Server Administration Guide.*

- Extending the Scope of Roles

Only scope your roles to different subtrees on the same server instance. Scoping roles to other servers may result in unpredictable behavior.

Deciding Between Groups and Roles

The groups and roles mechanisms provide some overlapping functionality that can lead to some ambiguity. Both methods of grouping entries have advantages and disadvantages. Generally speaking, the newer roles mechanism is designed to provide frequently required functionality in a more efficient manner. However, because the choice of a grouping mechanism influences your server complexity and determines how your client processes membership information, you need to plan your grouping mechanism carefully. You must be sure to understand what type of set membership query and set management operations you will be needing to perform in order to decide which mechanism fits your needs. The following two parts in this section present the advantages and disadvantages of both mechanisms and should guide you in your design choice. We strongly recommend that you document your design choice to allow administrators to maintain the grouping policy consistently at a later date.

This section is divided into the following parts:

- Advantages of the Groups Mechanism
- Advantages of the Roles Mechanism

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, then it is less costly to use static groups, provided that the number of members does not exceed 20,000 as static groups with more than this number of members would have a negative impact on performance. Enumerating members of a static group by retrieving the `member` attribute is easier than recovering all entries that share a role, thus making them more suited to member enumeration operations.

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

  If you want to assign a user to a set or remove a user from a set the simplest grouping mechanism for performing this task is static groups because it does not involve having special access rights 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, which 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, then the groups mechanism is the simplest to use, as there are no nesting restrictions. The roles mechanism, on the other hand, only allows nested roles to receive other roles.

- Groups are preferable to roles if freedom of scope for grouping your entries is critical to your deployment.

  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 a given set *and* find all set membership for a given entry.

  The roles mechanism is your best choice for enumerating members of a given set *and* finding all set membership for a given entry, as roles push this 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 to this, all types of roles appear in this attribute, allowing the client to process all roles uniformly. Generally speaking roles can do 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 computation work that the server will automatically do regarding membership, then the roles mechanism is your best option. This is ultimately what roles were designed for, which means that they can be used in resource-orientated ACIs, as a basis for CoS, as part of more complex search filters, Password Policy, Account Inactivation, etc. Groups do not allow you to do this kind of integration work.

Managing Attributes with Class of Service (CoS)

As previously stated in the introduction to this section on grouping directory entries and managing attributes, the class of service (CoS) mechanism allows you to share attributes between entries in a way that is invisible 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, in that it does not group entries in the way that the roles mechanism does, but rather allows related entries share data for coherence and space considerations. This section examines the CoS mechanism in more detail and is divided into the following parts:

- About CoS
- CoS Definition Entries and Template Entries
- CoS Priorities
- Pointer, Indirect, 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 need to update each entry individually, a large job for administrators that is not only time consuming but also runs the risk of not updating all entries. 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. In addition, because there are less values actually 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.

The generated CoS attributes may be multi-valued from several templates. Specifiers may designate several template entries, or there may be several CoS definitions for the same attribute. Alternatively, you may specify template priorities so that only one value is generated from all chosen templates. For more information, refer to “Defining Class of Service (CoS)” in Chapter 5 of the *Sun ONE Directory Server Administration Guide.*

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

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

Finally, avoid changing CoS definitions more often than is strictly necessary. Modifications to CoS definitions do not take effect immediately, because the server caches CoS information. Although caching accelerates read access to generated attribute entries, when changes to CoS information actually occur, the server must reconstruct the cache, a task that takes 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 helper entries called the CoS definition entry and the CoS template entry. This section examines both entries in more detail and is divided into the following parts:

- CoS Definition Entry
- CoS Template Entry

CoS Definition Entry

The CoS definition entry identifies the type of CoS you are using and the names of the CoS attribute 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 deviations, 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, if needed. 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 It is interesting to note that 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 will be generated for the CoS attribute. All entries within the scope of the CoS will 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 will select 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 to make for easier management. It is also wise to name them in a way that suggests the functionality they provide. For example, a definition entry DN such as `"cn=classicCosGenerateEmployeeType,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 Chapter 5 of the *Sun ONE 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.

For example, a CoS template entry appears as follows:

```
dn: cn=exampleUS,cn=data
objectclass: top
objectclass: cosTemplate
postalCode: 44438
cosPriority: 0
```

This template entry contains the value for the `postalCode` attribute. It contains a priority of zero, which means it has precedence over any other conflicting templates that define a different `postalCode` value.

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 presented in more detail in this section are:

- Pointer CoS
- Indirect CoS
- Classic CoS

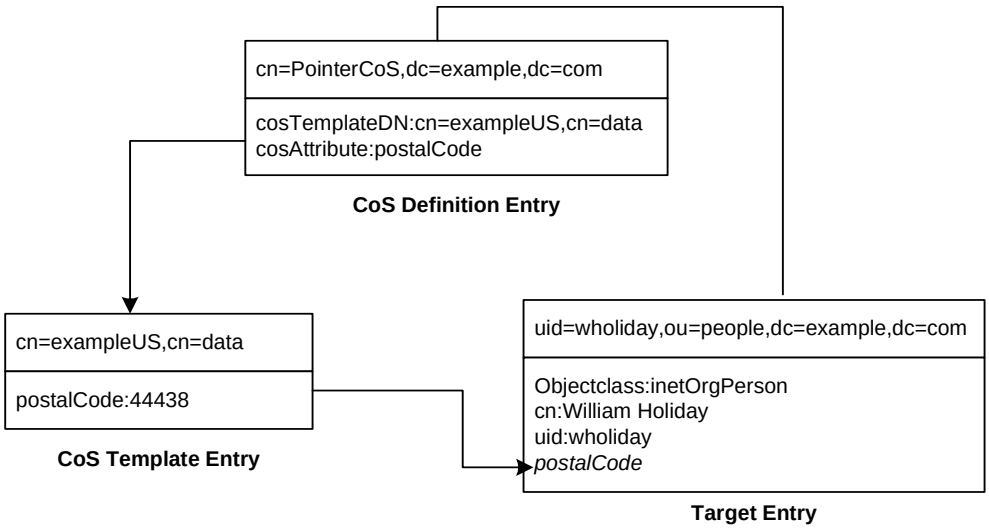
Pointer CoS

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

Pointer CoS Example

This example shows a CoS that defines a common postal code for all of the entries stored under `dc=example,dc=com`. The three entries (CoS definition entry, CoS template entry and the target entry) for this example are shown in Figure 4-10 on page 84:

Figure 4-10 Example of a Pointer CoS Definition and Template



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. When we imagine a scenario where several shared attributes are generated by CoS for thousands or millions or entries, instead of existing as real attributes in each entry, we appreciate the storage space savings and performance gains that CoS makes possible.

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.

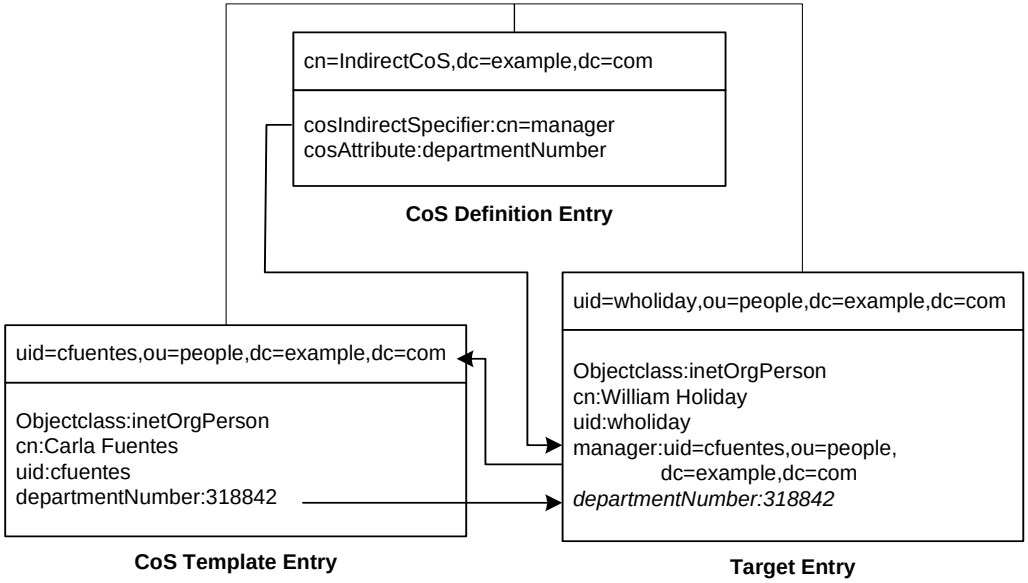
CAUTION Avoid overusing indirect CoS. Because templates may be arbitrary entries anywhere in the directory tree, implementing access control for indirect CoS can become extremely complex. In deployments where performance is critical, it is also wise to avoid overusing indirect CoS due to its resource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

This example of an indirect CoS 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. The three entries for this example are shown in Figure 4-11 on page 86:

Figure 4-11 Example of an Indirect CoS Definition and Template



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 `cn=Carla Fuentes,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 domain 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.

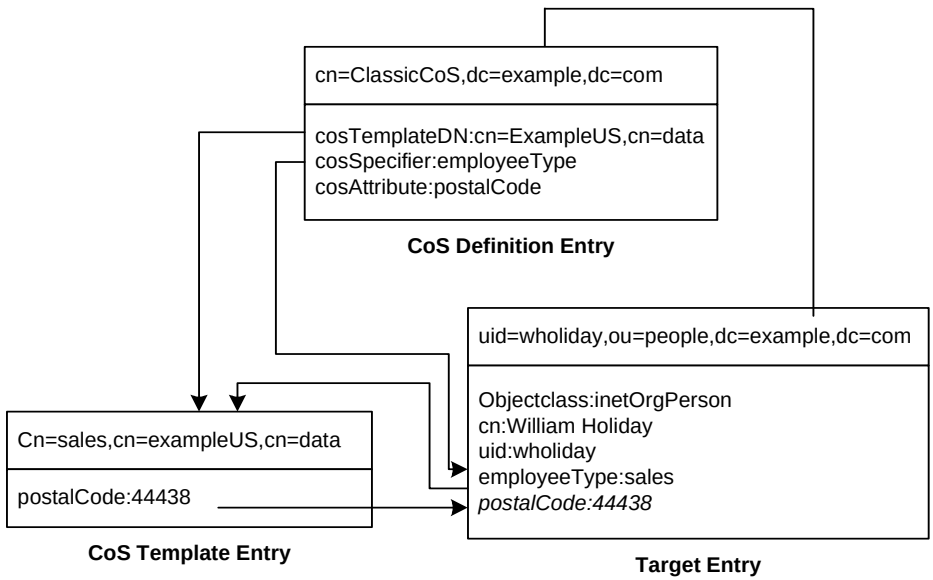
Classic CoS Example

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 will be taken as the cn value in the template DN. Template DNs for classic CoS must therefore have the following structure:

cn=*specifierValue*,*baseDN*

The example inFigure 4-12 on page 87 shows a classis CoS definition that generates a value for the postal code attribute:

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



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.

- Searches in suffixes where an attribute is declared as a CoS generated attribute will result in an unindexed search.

CoS generated attributes will only result in unindexed searches in suffixes where they are declared as CoS attributes. This may have a significant impact on performance. In suffixes where the same attribute is NOT declared as a CoS attribute, then the search will be indexed.

- Restricted Attribute Types

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

  - `userPassword` - A CoS-generated password value cannot be used to bind to the directory server.
  - `aci` - The directory server will not apply any access control based on the contents of a virtual ACI value defined by CoS.
  - `objectclass` - The 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 as described in the “CoS Limitations” section of the *Sun ONE 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 the "Creating Role-Based Attributes" section in the *Sun ONE 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 88.

- CoS Cache Latency

The CoS cache is an internal Directory Server structure which 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 upon 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. Bear this latency in mind before embarking on overly complex CoS configurations.

Directory Tree Design Examples

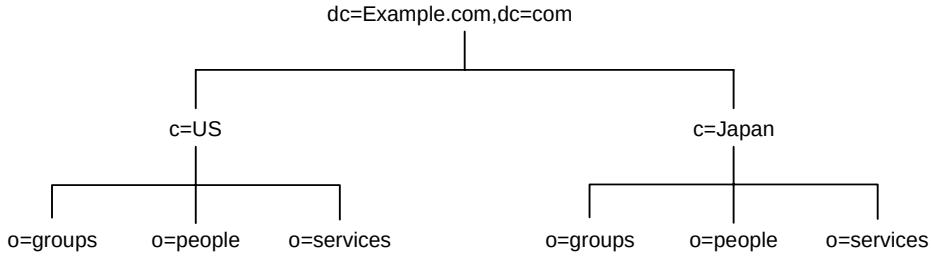
The following sections provide examples of directory trees designed to support a flat hierarchy as well as several examples of more complicated hierarchies.

Directory Tree for an International Enterprise

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

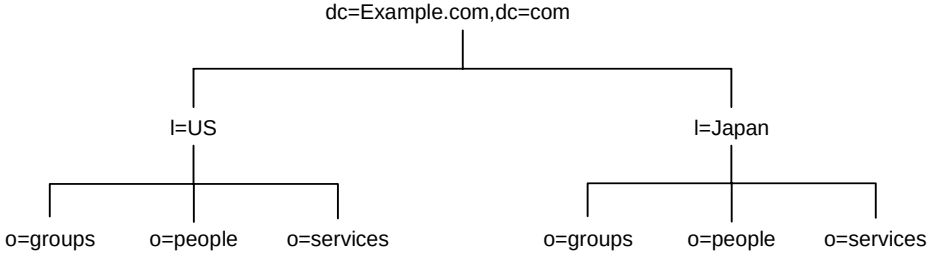
Because LDAP places no restrictions on the order of the attributes in your DNs, you can use the `c` (`countryName`) attribute to represent each country branch as illustrated in Figure 4-13 on page 90:

Figure 4-13 Use of c (countryName) Attribute to Represent Countries in a Directory Information tree



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

Figure 4-14 Use of l (locality) Attribute to Represent Countries in a Directory Information Tree



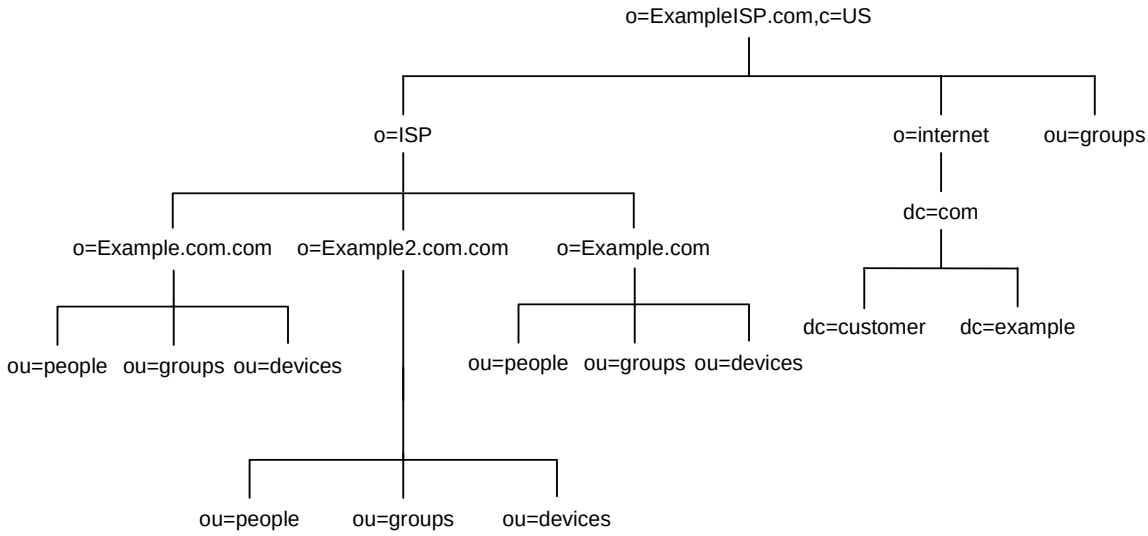
Directory Tree for an ISP

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

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

In addition, partitioning helps reduce performance problems caused by disk contention, and reduces the number of accounts potentially affected by a disk outage. Figure 4-15 on page 91 below shows the directory tree for Example.com, an ISP:

Figure 4-15 Directory Tree for Example.com ISP



Other Directory Tree Resources

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

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

Chapter 5

Designing the Directory Topology

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

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

- Topology Overview
- Distributing Your Data
- About Referrals and Chaining

Topology Overview

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

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

The database is the basic unit you use for jobs such as replication, performing backups, and restoring data.

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

Sun ONE Directory Server also provides the *referral* and *chaining* mechanisms for linking directory data stored in different databases.

The remainder of this chapter describes databases, referrals, and chaining, and describes how you can design indexes to improve the performance of your databases.

Distributing Your Data

Distributing your data allows you to scale your directory across multiple server instances which, may or may not, depending on your performance requirements, be stored on several machines, without physically containing those directory entries on each server in your enterprise. A distributed directory can thus hold a much larger number of entries than would be possible with a single server.

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

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

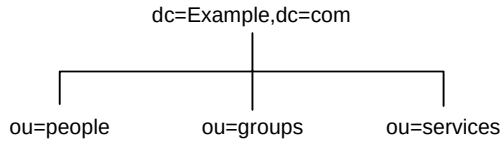
- Using Multiple Databases
- About Suffixes

Using Multiple Databases

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

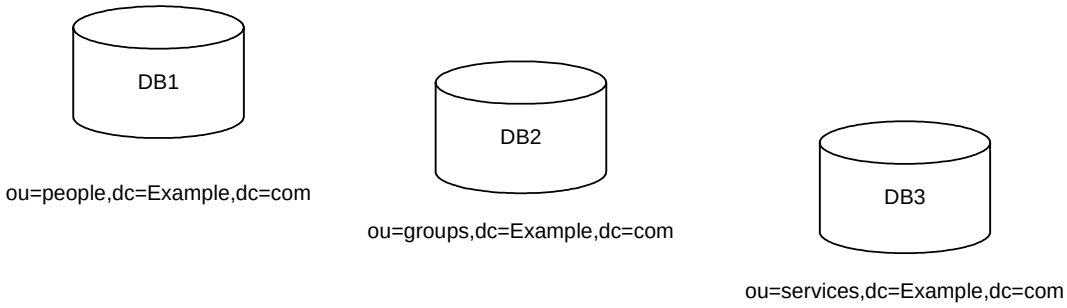
You can store different portions of your directory tree in different databases. For example, your directory tree appears as shown in Figure 5-1:

Figure 5-1 Directory Tree With Three Subsuffixes



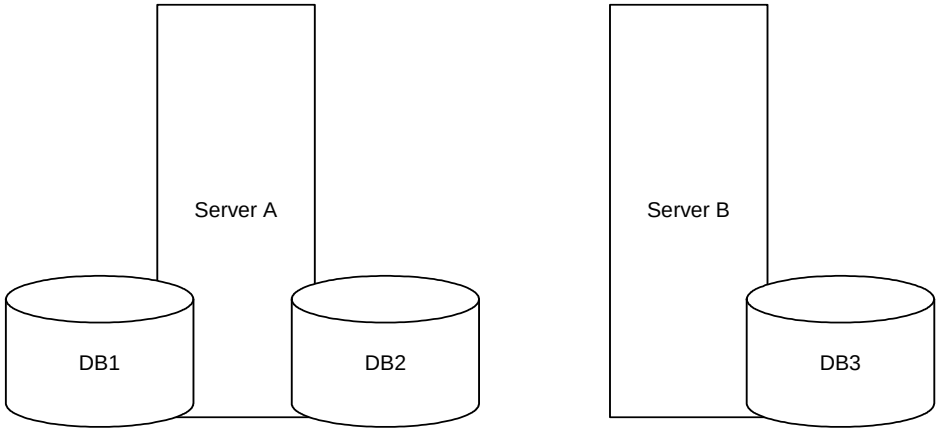
You can store the data of the three subsuffixes in three separate databases as shown in Figure 5-2:

Figure 5-2 Three subsuffixes Stored in Three Separate Databases



When you divide your directory tree among a number of databases, these databases can then be distributed across multiple servers, which generally equates to several physical machines to improve performance. For example, the three databases you created to contain the three subsuffixes of your directory tree can be stored on two servers as shown in Figure 5-1 on page 95:

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



Server A contains databases DB1 and DB2 and server B contains database DB3. Distributing databases across multiple servers reduces the amount of work each server needs to do. Thus, the directory can be made to scale to a much larger number of entries than would be possible with a single server.

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

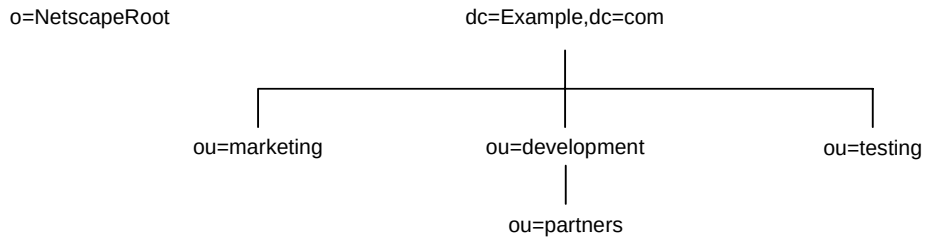
About Suffixes

Each database contains the data within a suffix of your directory server. You can create both 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 your directory tree or part of a larger tree you have designed for your directory server.

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

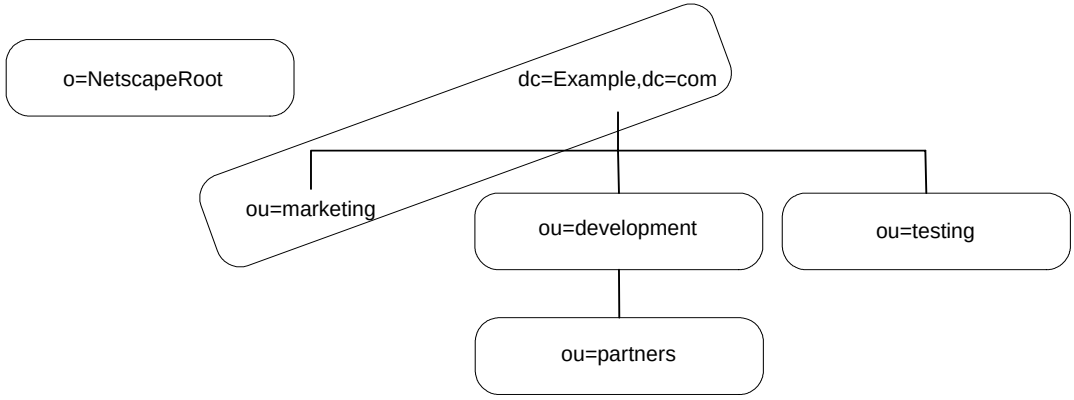
For example, you want to create subsuffixes to represent the distribution of your directory data. The directory tree for Example.com Corporation appears as shown in Figure 5-4 on page 97:

Figure 5-4 Example.com Corporation's Directory Tree



Example.com Corporation decides to split their directory tree across five different databases as illustrated in Figure 5-5 on page 97:

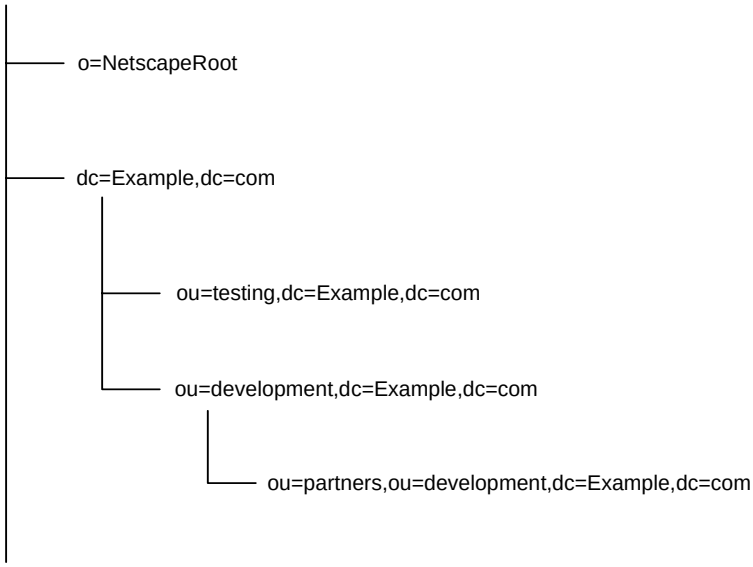
Figure 5-5 Example.com Corporation's Directory Tree Split Across Five Databases



`o=NetscapeRoot` and `dc=Example,dc=com` are both suffixes. The other `ou=testing,dc=Example,dc=com`, `ou=development,dc=Example,dc=com`, and `ou=partners,ou=development,dc=Example,dc=com` suffixes are all 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 on page 98:

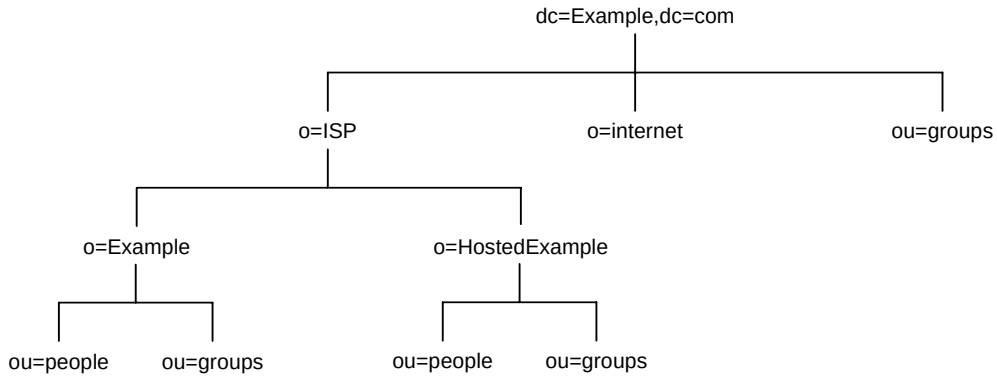
Figure 5-6 Example.com Corporation Suffixes and Associated Entries



Your directory might contain more than one suffix. For example, an ISP called Example.com might host several websites, one for its own website Example.com and one for another website called HostedExample2.com. The ISP can choose between creating one suffix, which houses everything, or two suffixes to separate the hosted ISP part of the organization from internal Example.com data.

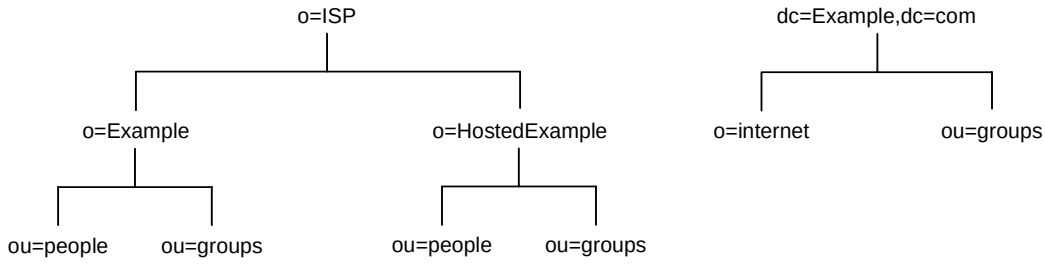
The first solution with just one suffix for all data, would have the directory information tree as shown in Figure 5-7 on page 99:

Figure 5-7 ISP Example.com Directory Tree with One Suffix



Whereas the second solution where the ISP creates two suffixes, one corresponding to the `dc=Example,dc=com` naming context and one corresponding to the `o=ISP` part of the organization would have the directory information tree as shown in Figure 5-8 on page 99:

Figure 5-8 ISP Example.com's Directory Tree



The `dc=Example,dc=com` entry represents a suffix as does the `o=ISP` entry. The entries for each hosted ISP, that is, `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 ISP.

About Referrals and Chaining

Once your data is distributed over several databases, you need to define the relationships between the distributed data. You do this using pointers to directory information held in different databases. The Sun ONE Directory Server provides the referral and chaining mechansims to help you link your distributed data into a single directory tree.

- Referrals

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

- Chaining

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

The following sections describe and compare these two mechanisms in more detail.

Using Referrals

A referral is a piece of information returned by a server that tells a client application the server to contact to proceed with an operation request. 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 server level using the `nsslapd-referral` attribute.

- Suffix Referrals

  A suffix referral, as opposed to a default referral, is a referral stored at database level. Suffix referrals are configured in the suffix configuration information. You can set a suffix referral for each suffix, which, generally speaking, means setting a suffix per database.

- Smart referrals

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

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

The Structure of an LDAP Referral

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

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

For example, a client application searches `dc=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 you use determines how a referral is handled. Some client applications automatically retry the operation on the server to which they have been referred. Other client applications simply return the referral information to the user. Most LDAP client applications provided by Sun ONE 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

The directory server determines whether a default referral should be returned by comparing the DN of the requested directory object against the directory suffixes supported by the local server. If the DN does not match the supported suffixes, the directory server returns a default referral.

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

`uid=bjensen,ou=people,dc=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 the Setting Default Referrals section in the *Sun ONE Directory Server Administration Guide.*

Suffix Referrals

Suffix referrals allow you to configure referrals at the suffix level. This functionality offers a more detailed level of referral and, with one of the configuration attributes, allows you to control where updates are actually made. Default referrals for each database in your directory installation are also set by the `nsslapd-referral` attribute in the mapping tree entry of the `dse.ldif` configuration file.

Imagine you have two major sites in the US, one based in New York and the other in Los Angeles. A client application sends 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 can be processed by the suffix which contains the `dc=NewYork` subtree.

Suffixes can be configured to operate in four different states, two of which concern suffix referrals. If you do not require suffix referrals you can choose between the `nsslapd-referral: backend` state, where the backend (database) processes all operations, and the `nsslapd-referral: disabled` state, where the database is not available for processing and an error is returned in response to requests made by the client applications.

However, if you want to configure suffix referrals, then you can choose to configure two different states. The `nsslapd-referral: referral` state causes a referral to be returned for all requests made to this suffix, and the `nsslapd-referral: referral on update` state causes the database to be used for all operations except update requests which will receive a referral. The second `referral on update` state is used internally by the server when replication is

configured to prevent consumers from processing update requests. However, should you need to restrict access to read operations on certain databases for load balancing or performance reasons, this suffix referral configuration possibility would be your solution.

For information on configuring suffix referrals, see the Creating Suffix Referrals section in the *Sun ONE Directory Server Administration Guide*. For more information concerning the configuration attributes

Smart Referrals

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

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

Unlike default referrals, the smart referrals are stored within the directory itself. For information on configuring and managing smart referrals, refer to the Creating Smart Referrals section in the *Sun ONE Directory Server Administration Guide*.

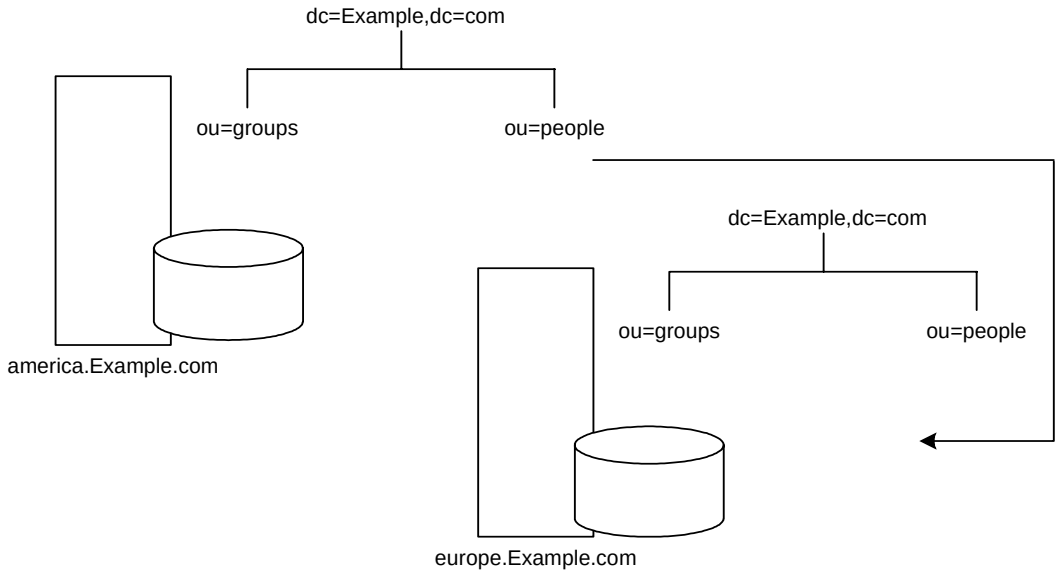
For example, the directory for the American office of Example.com Corporation contains the following directory branch point: `ou=people,dc=Example,dc=com`.

You redirect all requests on this branch to the `ou=people` branch of the European office of Example.com Corporation 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 where requests made to the American directory are redirected to the European directory is shown in Figure 5-9 on page 104:

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

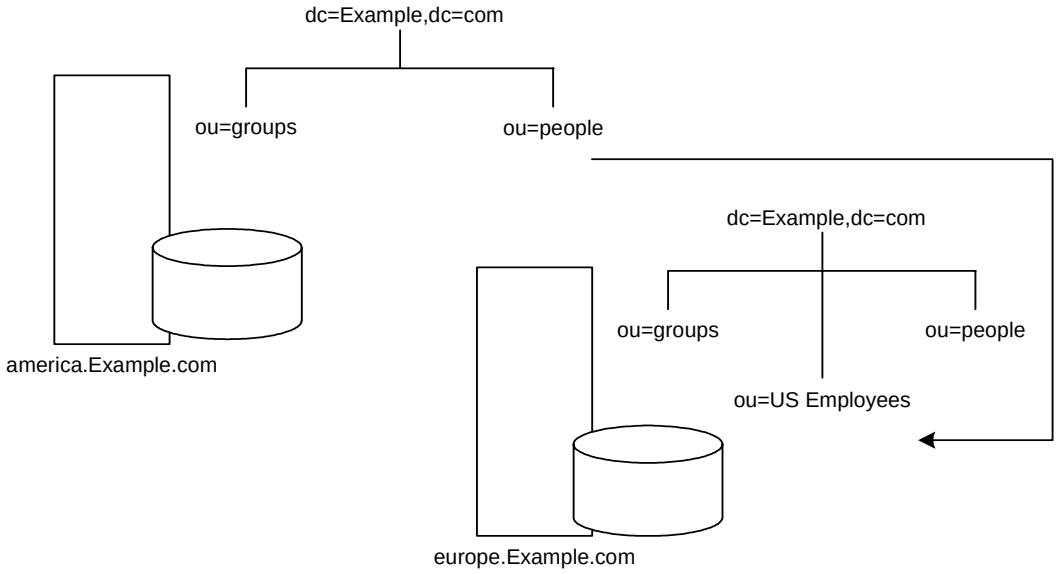


You can use the same mechanism to redirect queries to a different server that uses a different namespace. For example, an employee working in the Italian office of Example.com Corporation makes a request to the European directory for the phone number of a Example.com employee in America. The referral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

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

Figure 5-10 on page 105 illustrates the smart referral for an Italian employee in the European office requesting the phone number of an American employee in the American office.

Figure 5-10 Smart Referral for Phone Number Request

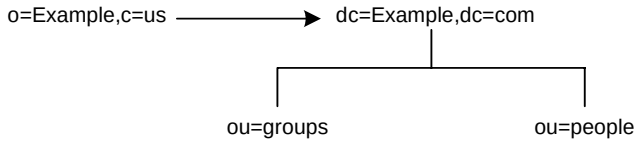


Finally, if you serve multiple suffixes on the same server, you can redirect queries from one namespace to another namespace served on the same machine. If you want to redirect all queries on the local machine for `o=Example,c=us` to `dc=Example,dc=com`, then you would put the following smart referral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

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

as illustrated in Figure 5-11:

Figure 5-11 Smart Referral Traffic



Since 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 after the second slash. As a result, 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 the LDAP URLs appendix in the *Sun ONE Directory Server Reference Manual*. For more information on how to include smart URLs on Sun ONE Directory Server entries, see the Setting Referrals section in the *Sun ONE Directory Server Administration Guide*.

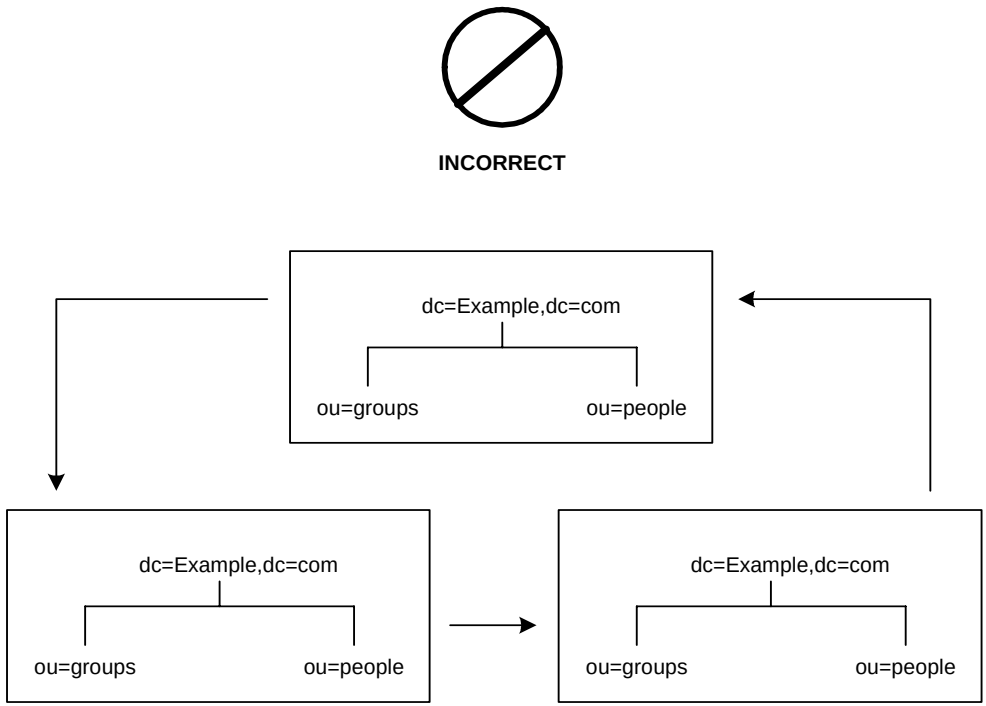
Tips for Designing Smart Referrals

Consider the following points before using smart referrals:

- Keep the design simple.

  Deploying your directory using a complex web of referrals makes administration difficult. Also, overusing smart referrals can lead to circular referral patterns. For example, a referral points to an LDAP URL, this LDAP URL in turn points to another LDAP URL, and so on until a referral somewhere in the chain points back to the original server. A circular referral pattern is depicted in Figure 5-12:

Figure 5-12 Incorrect Circular Referral Pattern Caused by the Overuse of Smart Referrals



- Redirect at major branch points.

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

- Consider the security implications.

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

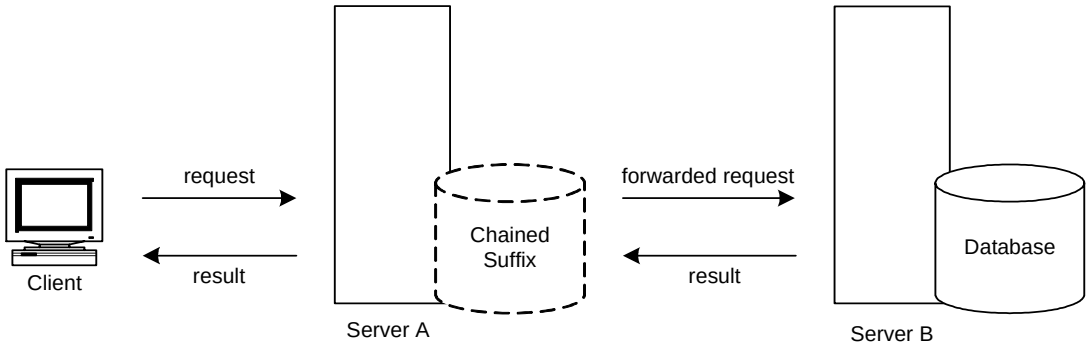
  Also, the client credentials need to be available on the server to which the client is referred for client authentication to take place.

Using Chaining

Chaining is a method for relaying requests to another server. This method is implemented through chained suffixes. A chained suffix, as described in the section titled "Distributing Your Data", 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 5-13 on page 109.

Figure 5-13 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 the Creating Chained Suffixes section in the *Sun ONE 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 the Access Control Through Chained Suffixes section in the *Sun ONE Directory Server Administration Guide*.

Deciding Between Referrals and Chaining

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

The 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.

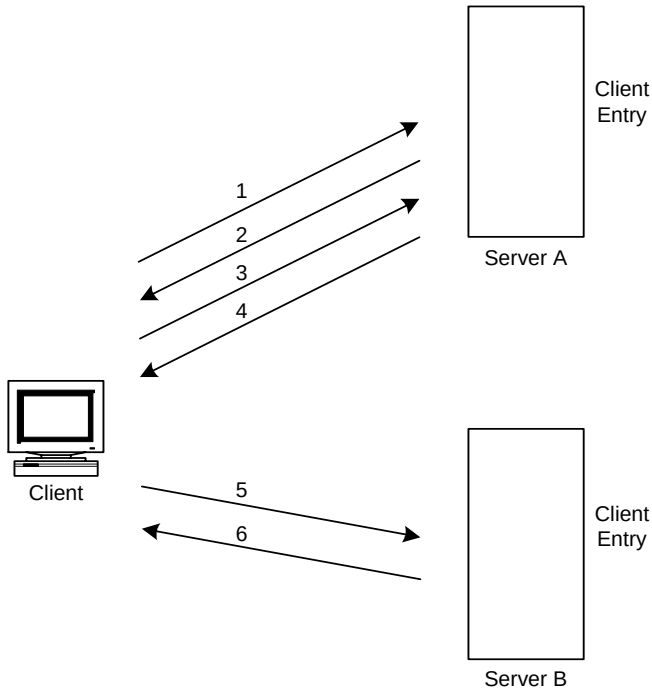
Also, with referrals a client must authenticate, meaning that the servers to which clients are being referred need to contain the client credentials. With chaining, client authentication takes place only once. Clients do not need to authenticate again on the servers to which their requests are chained.

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 5-14 on page 111 shows how the operation would work using referrals:

Figure 5-14 Client Application Search Request to Server A Being Redirected to Server B Through a Referral



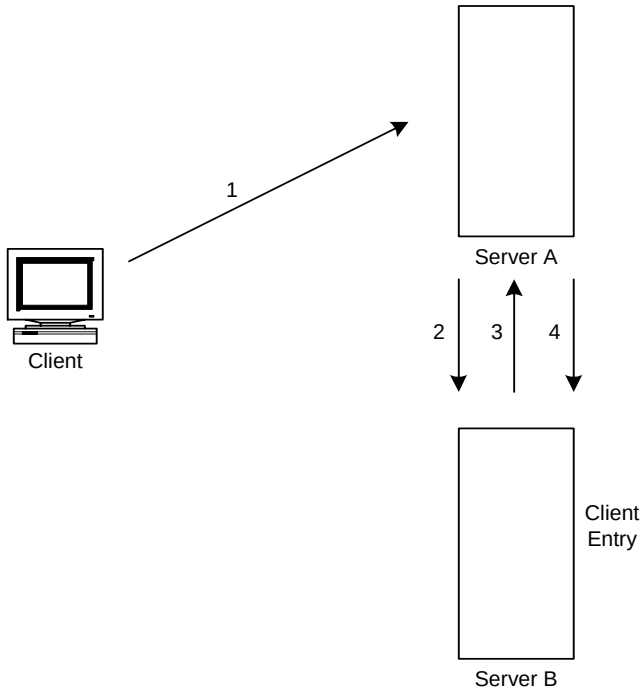
In the illustration above, the client application performs the following steps:

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

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

Chaining solves this problem. A search request using chaining would work as shown in Figure 5-15 on page 112:

Figure 5-15 Search Request using Chaining



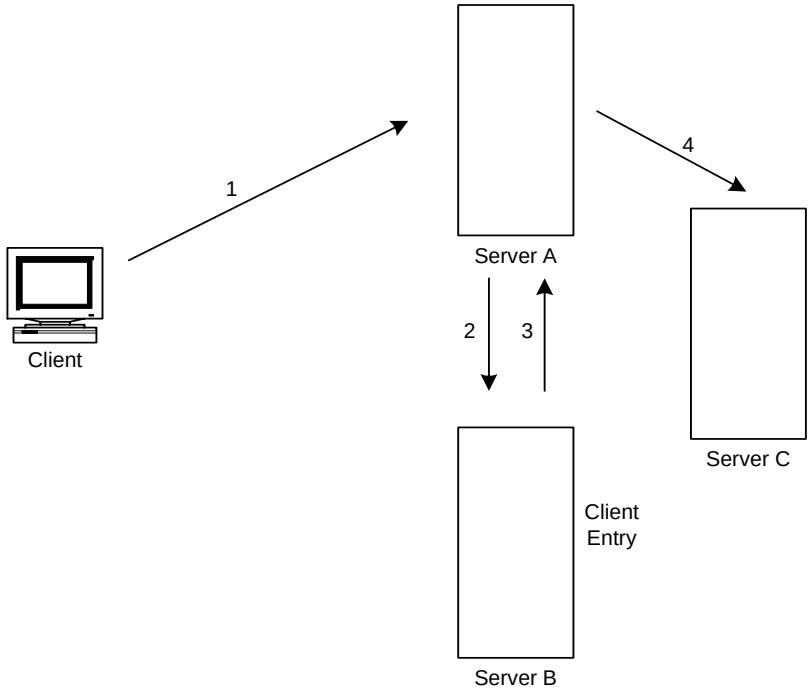
In the illustration above, the following steps are performed:

1. The client application binds with Server A and Server A tries to confirm that the user name and password are correct.
2. Server A does not contain an entry corresponding to the client application. Instead, it contains a 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 5-16 on page 113 illustrates how two chained suffixes can be used to satisfy a client's search request:

Figure 5-16 Chaining Using Two Chained Suffixes to Process a Client's Search Request



In this illustration, the following steps are performed:

1. The client application binds with Server A and Server A tries to confirm that the user name and password are correct.
2. Server A does not contain an entry corresponding to the client application. Instead, it contains a 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

Designing the Replication Process

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

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

- Introduction to Replication
- Common Replication Scenarios
- Defining a Replication Strategy
- Using Replication with Other Directory Features
- 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 database) between servers, except the configuration or monitoring information subtrees.

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

- Fault tolerance / Failover

By replicating directory trees to multiple servers, you can ensure your directory is available even if some hardware, software, or network problem prevents your directory client applications from accessing a particular Directory Server. Your clients 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 your replication environment.

- Load balancing

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

- Higher performance and reduced response times

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

- Local data management

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

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

- Replication Concepts
- Data Consistency

Replication Concepts

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

- What information you want to replicate.
- Which server or servers hold the master copy of that information.
- Which server or servers hold the read-only copy of the information.
- What should happen when a consumer replica receives modification requests from client applications; that is, to which server should it refer the request.

These decisions cannot be made effectively without an understanding of how the Directory Server handles these concepts. For example, when you decide what information you want to replicate, you need to know what is the smallest replication unit that the Directory Server can handle.

To ensure that you fully understand the replication process and the possibilities it provides you for your Directory Server deployment, the following sections explain the replication concepts used by Directory Server. This provides a solid framework for thinking about the global decisions you will need to take.

Replica

A database that participates in replication is defined as a *replica*. There are several 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 just like a consumer replica. The difference is that it is stored on a Directory Server that acts as a supplier of one or more consumer replicas.

You can configure a Directory Server to manage several replicas. Each replica can have a different role in replication. For example, you could have a Directory Server that stores the `dc=engineering,dc=example,dc=com` suffix in a master replica, and the `dc=sales,dc=example,dc=com` suffix in a consumer replica.

Unit of Replication

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

Replica ID

Master replicas require a unique replica identifier (ID) and consumer replicas all have the same replica ID. The replica ID for masters can be any 16 bit integer between 1 and 65534, while consumer replicas all have the replica ID of 65535. The replica ID lies at the heart of the replication mechanism as it identifies to which replica the changes occurred, thus enabling them to be replicated correctly.

NOTE If a server hosts several replicas, the replicas may have the same replica ID, provided that the replica ID is unique between the masters of a single, replicated naming context or suffix.

Supplier/Consumer

A server that replicates to other servers is called a *supplier*. A server that is updated by other servers is called a *consumer*.

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

- When the Directory Server manages a combination of master replicas and consumer replicas.
- When the Directory 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 134.
- In multi-master replication, when a master replica is mastered on two different Directory Servers, each Directory Server acts as a supplier and a consumer of the other Directory Server. For more information, refer to "Multi-Master Replication," on page 127.

When we refer to a server that only plays the role of consumer; that is, it only contains a consumer replica, we refer to this server as a dedicated consumer.

In Directory Server, replication is always initiated by the supplier, never by the consumer. We refer to this as supplier-initiated replication, as suppliers push the data to consumers.

Earlier versions of the Directory Server allowed consumer-initiated replication where you could configure consumers to pull data from a suppliers. Since the 5.0 release of Directory Server, 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, modrdn, or modify) 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 always 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 server that contains the master replica.
- Maintain the historical information for the replica.
- Initiate replication to consumers.

For more information on cascading replication, refer to “Cascading Replication,” on page 134.

For a consumer replica, the server must:

- Respond to read requests.
- Maintain historical information for the replica.
- Refer update requests to the server that contains the master replica.

  Anytime 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 the consumer or hub replicas not to return a referral, but to return an error instead if it is desirable for security and performance reasons to do so. Refer to the Note on page 131 for more information.

Online Replica Promotion and Demotion

Sun ONE Directory Server 5.2 provides online replica promotion and demotion functionality. Once online promotion or demotion is complete, the servers immediately start or stop accepting updates. 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.

In addition to providing increased flexibility, online replica promotion and demotion affords you increased failover capabilities. Take the example of a two-way, multi-master scenario with two hubs configured for additional load balancing and failover. Should one of the masters go offline, you simply need to promote one of the hubs to maintain optimal read-write availability, and then, when the master replica comes back online, a simple demotion back to hub replica returns you to the original state of affairs.

NOTE Before demoting a hub to a consumer, which will result in the replica no longer being able to propagate any changes due to the fact that as a consumer it will not have a change log, you must verify that the hub is in sync with the other servers. To ensure that the hub is in sync you can use the replication monitoring tool `insync`, which is presented in the section entitled "Replication Monitoring," on page 159.

Change Log

Every server acting as a supplier, that is 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 server acting as a supplier then 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 need to use the Retro Change Log Plug-in for backward compatibility. For more information about the Retro Change Log Plug-in refer to the Using the Retro Change Log Plug-In section in the *Sun ONE Directory Server Administration Guide*.

NOTE Care should be taken when planning the change log size because once entries are purged from the change log, they can no longer be replicated. You need to consider carefully the type of traffic you expect to be sure to provide sufficient change log disk space as different types of changes require different amounts of disk space.

Replication Identity

When replication occurs between two servers, the server acting as the consumer authenticates the server acting as supplier when it 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, the Directory Server Console refers to DN or bind DN, it is referring to the bind DN of the Replication Manager Entry.

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

- You must have at least one on every server acting as a consumer (whether they be dedicated consumers, hubs, or masters in a multi-master environment).
- This entry must not be part of the replicated data for security reasons and initialization issues.

NOTE This entry has a special user profile that bypasses all access control rules defined on the consumer server. However, 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 server acting as the consumer, 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) in your replication topology.
- On the server acting as the supplier, that is all master and hub replicas, when you configure the replication agreement, you must specify the bind DN of this entry in the replication agreement.

NOTE In the Directory Server Console, this Replication Manager entry is created by default, although the Directory Server Console does allow you to create your own should you so desire.

If you are using SSL and replication and want to authenticate then there are two possible methods:

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

Replication Agreement

Directory Servers use replication agreements to define replication. A replication agreement describes replication between *one* supplier and *one* consumer. The agreement is configured on the supplier. For replication to work it is important to remember that the replication agreement must be enabled. It identifies:

- The database to replicate.
- The consumer server to which the data is pushed.
- A pointer to a set of attributes to exclude or include from the replicated data if fractional replication is configured.
- The times during which replication can occur.
- The bind DN and credentials the supplier must use to bind to the consumer, called the Replication Manager entry (for more information, refer to "Replication Identity," on page 120).
- How the connection is secured (SSL, client authentication).
- 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.
- Status information about the replication agreement.
- On Solaris and Linux systems, information on the level of compression used in replication.

NOTE In Sun ONE Directory Server 5.2 you can choose to disable or enable existing replication agreements. This can be useful should you temporarily have no need to use a particular replication agreement, but want to maintain its configuration for possible future use.

Consumer Initialization or Total Update

Consumer initialization, or total update, is the process whereby you physically copy all data from the server acting as the supplier to the server acting as the consumer. Once you have created a replication agreement, the consumer within that agreement needs to be initialized. It is only after consumer initialization is complete, that the supplier can begin replaying, or replicating the future update operations to the consumer(s). Under normal operations, the consumer should not require further initialization; however, if the data on a supplier is restored from backup for any reason, then you may need to re-initialize some of the consumers

dependent on that supplier. An example where consumer re-initialization would be necessary is if the restored supplier was the only supplier for the consumer in the topology. It is possible to initialize consumers both online and offline (manually). For further information on the consumer initialization procedures see the Initializing Replicas section in the *Sun ONE Directory Server Administration Guide*. Directory Server 5.2 also offers an advanced binary copy feature which can be used to clone either master or consumer replicas using the binary backup files from 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 very large database files. For information on the binary backup procedures and an exhaustive list of the feature's strict limitations see "Binary Backup (db2bak)," on page 265.

In a multi-master replication topology, the default behavior of a read-write replica that has been reinitialized either online or offline from a backup or an ldif file, is to REFUSE client update requests. Note that this is in contrast to previous versions of Directory Server. By default the replica will remain in read-only mode indefinitely and will refer any update operations to other suppliers in the topology. In such a case, the administrator may configure the replica to begin accepting updates again in two ways:

- Manually enable read-write mode by using the Directory Server console or setting the `ds5BeginReplicaAcceptUpdates` attribute to `start`. This allows the administrator to use the `insync` replication monitoring tool to ensure that the replica has completely converged with the other suppliers in the topology. This is the recommended procedure because the administrator can guarantee that the replica is in sync before allowing update operations.
- Configure the replica to automatically revert to read-write mode after a given delay specified by the replica specific `ds5referralDelayAfterInit` attribute. This procedure presents the risk of allowing update operations on the replica before it is completely synchronized with the other master replicas, which may lead to unexpected errors.

For more information on these procedures refer to the the Initializing Replicas section of the *Sun ONE Directory Server Administration Guide*. For more information regarding the replication configuration attributes refer to the replication attributes listed in the Core Server Configuration Attributes chapter of the *Sun ONE Directory Server Reference Manual*.

Incremental Update

Incremental update is the process whereby updates are replicated by the supplier to the consumer following consumer initialization or total update. In contrast to previous releases of Directory Server, Sun ONE Directory Server 5.2 allows a consumer to be incrementally updated by several suppliers at once, provided that the updates themselves originate from different replica IDs. Simultaneous incremental updates from several suppliers (but different replica IDs) improves the performance of the incremental update procedure.

Data Consistency

Consistency refers to how closely the contents of replicated databases match each other at a given point in time. When you set up replication between two servers, part of the configuration is to schedule updates. With Directory Server, it is always the server acting as the supplier that determines when consumers need to be updated, and initiates replication. Replication can take place only after the consumers have been initialized.

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

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

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

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

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

- There is a latency in the propagation of 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 Scenarios

You need to decide how the updates flow from server to server and how the servers interact when propagating replication updates to build a replication strategy which fits your replication requirements. There are five basic scenarios:

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

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

NOTE Whatever replication scenario you choose to implement, remember to consider schema replication. See “Schema Replication,” on page 157 for further information.

Single-Master Replication

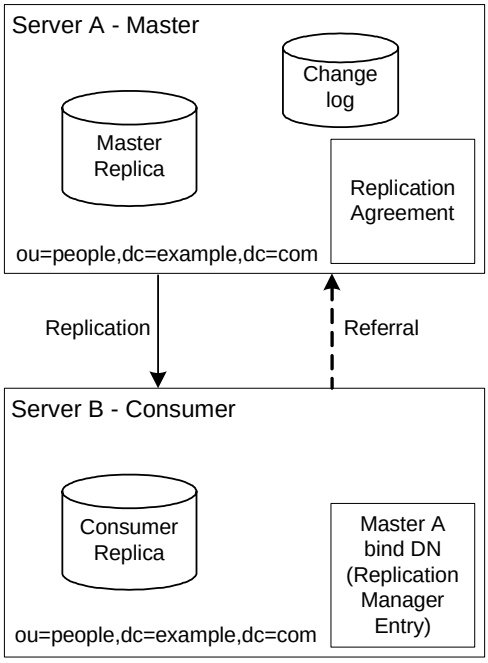
In the most basic replication configuration, a server acting as a supplier copies a master replica directly to one or more consumer servers. In this configuration, all directory modifications are made to the master replica stored on the supplier, and the consumers contain read-only copies of the data.

The supplier maintains a change log that records all the changes made to the master replica. The supplier also stores 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 server must propagate all modifications to the consumer replicas. Figure 6-1 on page 126 shows this simple configuration.

Figure 6-1 Single-Master Replication



In the example illustrated in Figure 6-1, the `ou=people,dc=example,dc=com` suffix receives a large number of search and update requests from clients. Therefore, 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.

NOTE In replication, the server acting as the consumer stores referral information about the server acting as the supplier, but does not forward modification requests from clients to the supplier. The client must follow the referral sent back by the consumer.

Although Figure 6-1 shows just one server acting as a consumer, the 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 environment, master replicas of the same information exist on more than one server. This section on multi-master replication is divided into the following parts:

- Multi-Master Replication Basic Concepts
- Multi-Master Replication Capabilities
- Fully-Connected, Four-Way, Multi-Master Topology
- Multi-Master Replication over Wide Area Networks (WAN)

Multi-Master Replication Basic Concepts

In a multi-master configuration where master replicas of the same information exist on more than one server, data can be updated simultaneously in two or more different locations. This means that each server maintains a change log for the master replica involved in the replication topology. The changes that occur on each server are replicated to the other(s). This means that each server plays both roles of supplier and consumer. Multi-master configurations have the following advantages:

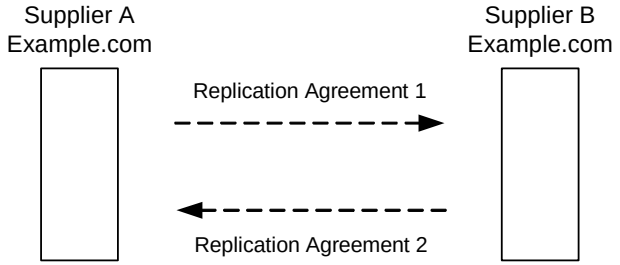
- Automatic write failover when one supplier is inaccessible.
- Updates are made on a local supplier in a geographically distributed environment.

When the same data is modified on both servers at approximately the same time, update reconciliation procedures are applied; i.e. the most recent change takes precedence. However, some conflicting changes may break the LDAP model, which will result in the entry being marked as a conflicting entry. To resolve these “conflicting entries,” the Administrator(s) will need to decide what to do with these entries and manually update them.

NOTE If the uniqueness of your attributes is important to your deployment, then we higly recommend that you enable the Attribute Value Uniqueness plug-in in multi-master replication environments, as it allows you to reduce the number of naming conflicts. For more information on the Attribute Value Uniqueness plug-in see the section entitled “Common Replication Scenarios,” on page 125.

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. So, to create a multi-master environment between two suppliers that share responsibility for the same data, you need to create two replication agreements. Figure 6-1 on page 126 shows this configuration:

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



Supplier A and Supplier B each hold a master replica of the same data and there are two replication agreements governing the replication flow of this multi-master configuration.

Directory Server 5.2 supports a maximum of four masters in a multi-master replication topology. The number of consumers 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

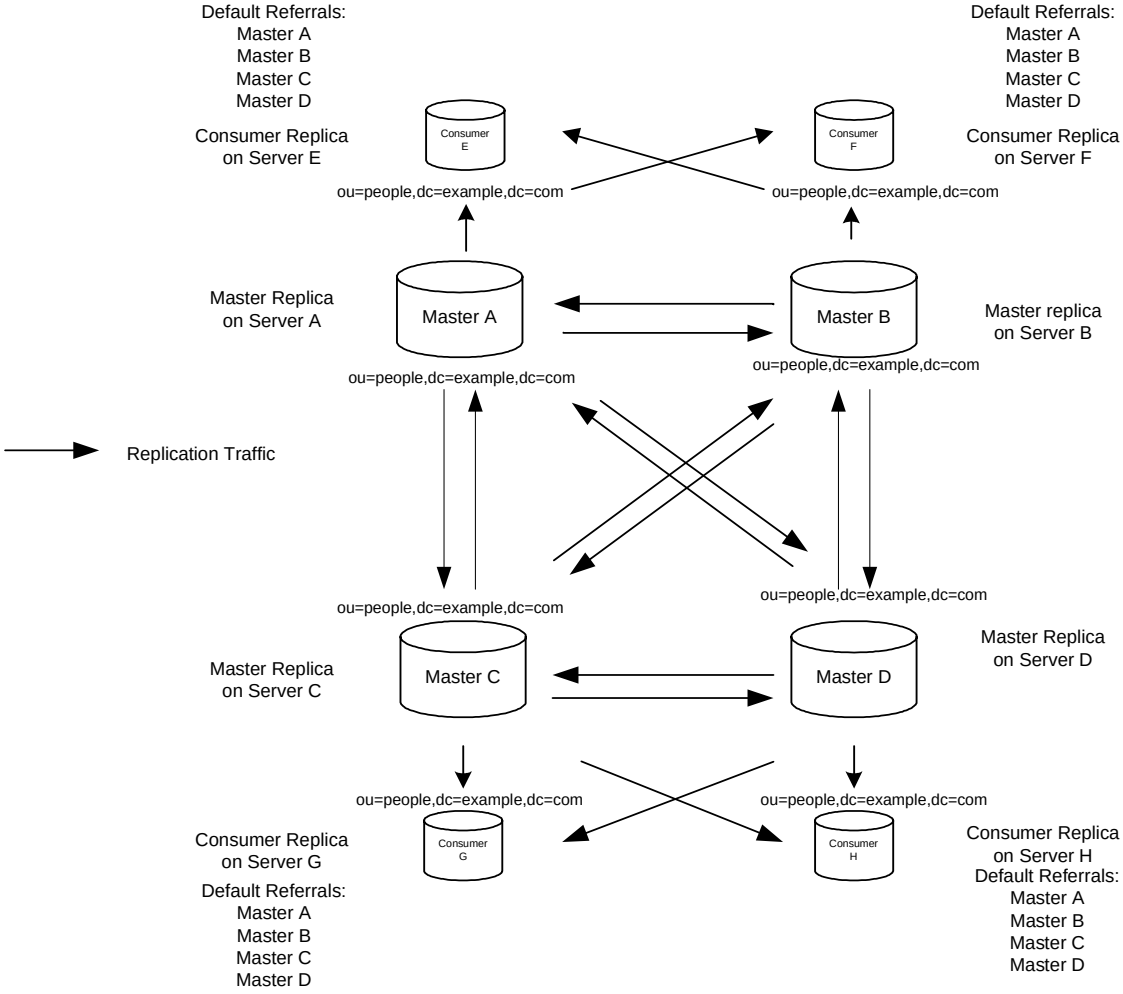
Sun ONE Directory Server 5.2 provides a more streamlined, flexible protocol that makes it easier for you to adapt your deployment to your replication and performance requirements. Sun ONE Directory Server 5.2 allows you to:

- Replicate updates based on the replica ID. Replica ID-based updates result in improved performance in that 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 with a given consumer, which provides you with greater replication configuration flexibility for your deployment. You can configure certain topologies in the knowledge that should you, at a later date, wish to modify that topology, you can easily do so.

Fully-Connected, Four-Way, Multi-Master Topology

Figure 6-3 on page 130 shows a fully-connected, four-way, multi-master topology. Thanks to its four-way master failover configuration, this fully-connected topology provides a highly-available solution that guarantees data integrity. It is the most secure in terms of read-write failover capability, but it is worth noting that this failover capability does not come without overheads in terms of performance. It will depend on your high-availability requirements as to whether or not you will want to deploy the fully-connected, multi-master configuration. Should your high-availability requirements be less stringent, or should you wish to reduce your replication traffic for performance reasons, you may want to opt for a “lighter” deployment in terms of read-write failover.

Figure 6-3 Fully-Connected Four-Way Multi-Master Replication Configuration



In Figure 6-3 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.

In Figure 6-3 each consumer stores two 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.

NOTE In replication environments consumers *do not* forward modification requests from clients to the servers acting as suppliers. In the event of a consumer receiving a modification request, the consumer will return a list containing the URLs of the possible masters that could satisfy the client's modification request.

Sun ONE Directory Server 5.2 allows you to control these referrals in that you can overwrite the referrals set automatically by the server by adding your own.

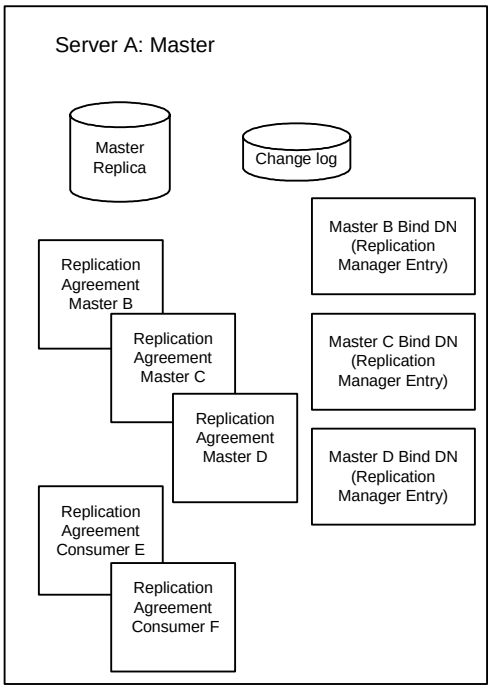
Being able to control referrals helps you to optimize your deployment's security and performance, in that it:

- ensures your referrals point to secure ports only,
- allows you to point to a Sun ONE Directory Proxy Server for load balancing reasons,
- allows you to redirect to a local server only in the case of a deployment with servers separated by a WAN, and
- allows you to limit referrals to a subset of masters in 4-way multi-master topologies.

For information regarding the configuration of referrals see the Setting Referrals section of the *Sun ONE Directory Server Administration Guide.*

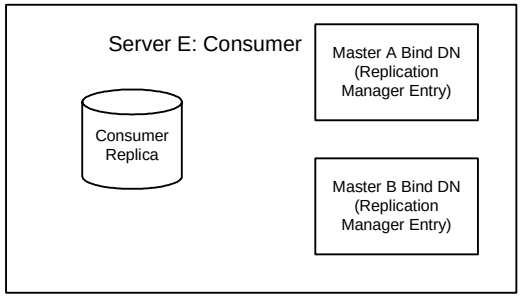
To better understand the replication elements you need to configure to deploy this fully-connected, four-way, multi-master replication topology, Figure 6-4 presents a detailed view of the replication agreements, change logs, and Replication Manager entries that you need to set up on master A, and Figure 6-5 provides the same detailed view for consumer E.

Figure 6-4 Replication Configuration for Master A in the Fully-Connected, Four-Way, Multi-Master Replication Topology



As we can see in Figure 6-4 the master A requires a master replica, a change log and Replication Manager entries or bind DNs for masters B, C, and D (in the case where you do not use the same Replication Manager entry for all four masters). In addition to the change log and Replication Manager entries, master A also requires replication agreements for the three other masters B, C, and D, and consumers E and F.

Figure 6-5 Replication Configuration for Consumer Server E in Fully-Connected, Four-Way, Multi-Master Replication Topology



The detailed view of the replication configuration for consumer E presented in Figure 6-5 shows us 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.

Multi-Master Replication over Wide Area Networks (WAN)

Multi-master replication (MMR) over Wide Area Networks (WAN) is a new feature of Sun ONE Directory Server 5.2 that will allow for MMR configurations across geographical boundaries in international, multiple data-center deployments. Previously master Directory Servers had to be connected via high-speed, low-latency networks with minimum connection speeds of 100Mb/second, for full MMR support which ruled out the possibility of MMR over WAN, but this is no longer the case. Sun ONE Directory Server now supports MMR over WAN, that means that geographical boundaries no longer constitute a stumbling block for multi-master replication. The flexibility this new feature will afford large deployments is immense.

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

In order to render MMR over WAN a viable deployment possibility, the Sun ONE Directory Server 5.2 replication protocol now provides for fully asynchronous support and window and grouping mechanisms. The following section will examine these mechanisms in more detail.

NOTE Although the viability of MMR over WAN is a direct result of these protocol improvements, they are equally valid for Local Area Network (LAN) deployments.

Grouping and Window Mechanisms

To optimize the replication flow, Directory Server allows you to group changes rather than having to send 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. You use the `ds5ReplicaTransportGroupSize` attribute to specify the number of changes that can be grouped into a single update request and the `ds5ReplicaTransportWindowSize` attribute to specify the number of sendUpdate requests that can occur before consumer acknowledgement is required. The default group size is 1 and the default window size is 10, which means that unless otherwise specified, default replication behavior will not group requests, but will allow 10 sendUpdate requests to be sent before consumer acknowledgement is required.

CAUTION Since both the grouping and window mechanisms are based on entry size, optimizing replication performance is difficult to configure when you have a variable entry size. If you are aware of a relatively constant entry size, you can use the grouping and window mechanisms to optimize incremental and total updates. It is also important to realize that the performance of your MMR over WAN replication traffic flow will depend the latency and bandwidth of your WAN connection.

You will need to analyze all of these factors carefully before configuring MMR over the WAN.

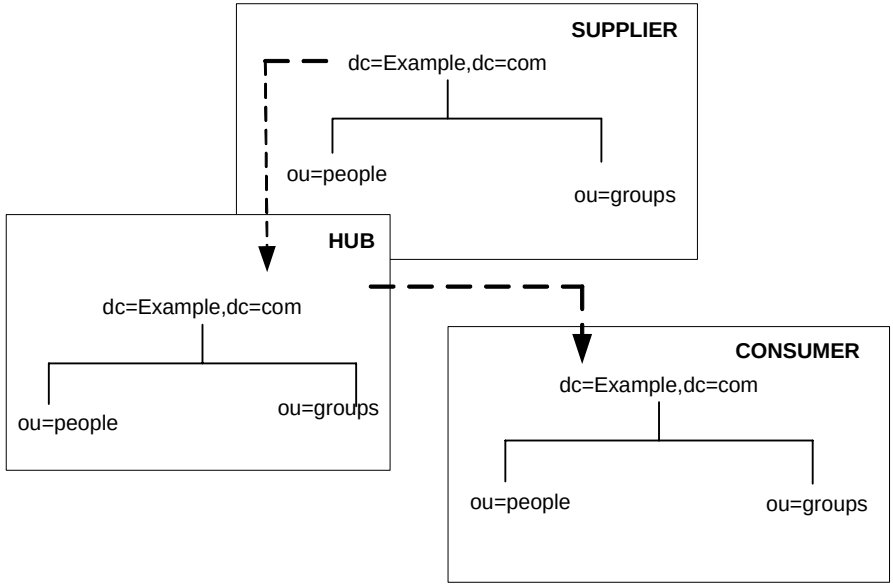
Cascading Replication

In a cascading replication scenario, a server acting as a *hub* receives updates from a server acting as a supplier, and replays those updates on 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 master refer update requests from directory clients to the master.

This cascading replication scenario is illustrated in Figure 6-6:

Figure 6-6 Cascading Replication Scenario

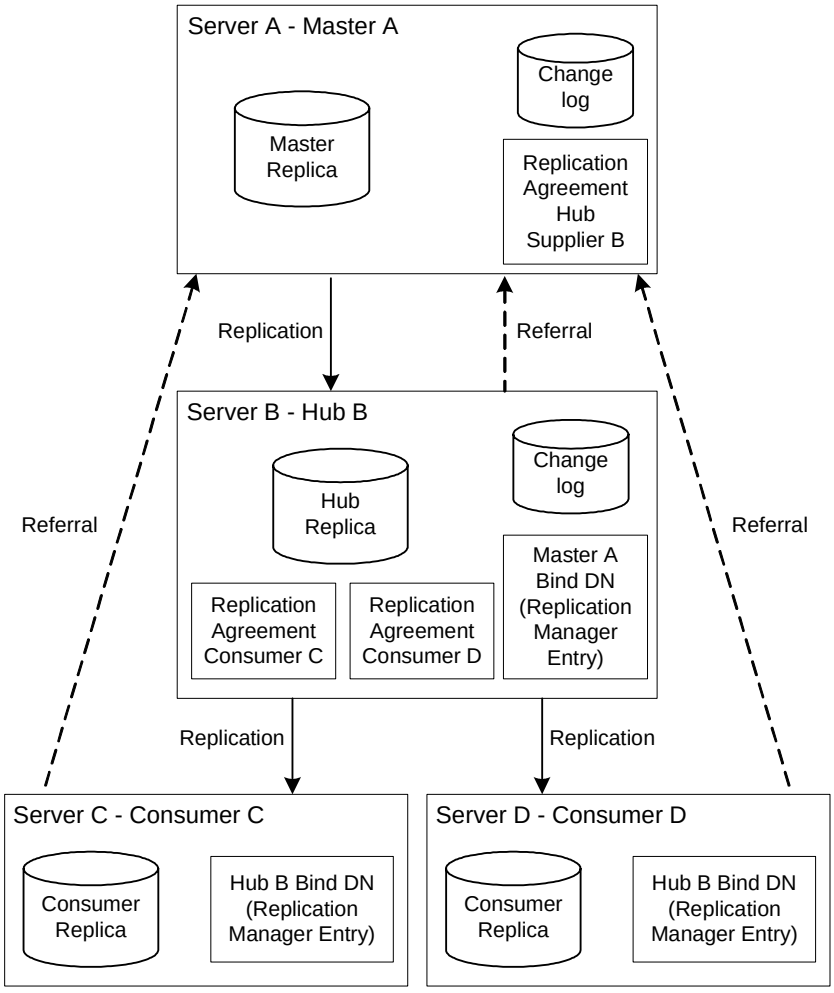


Cascading replication is very useful in the following cases:

- When you need to balance heavy traffic loads: for example, because your masters need to handle all update traffic, it would put them under a very heavy load to support all 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 dramatically increase the speed of replication between the server acting as the master and the server acting as the hub.

A similar scenario, from a different perspective, is illustrated in Figure 6-7. This illustration shows how the servers are configured in terms of Replication Agreements and change logs as well the default referrals.

Figure 6-7 Server Configuration in Cascading Replication



In the example illustrated in Figure 6-7, a server acting as a hub is used to balance the load of replication updates by sharing it between a server acting as a master server and the hub.

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, and consumers C and D both contain Replication Manager entries for Hub B, which it uses to authenticate when sending its updates to the consumers.

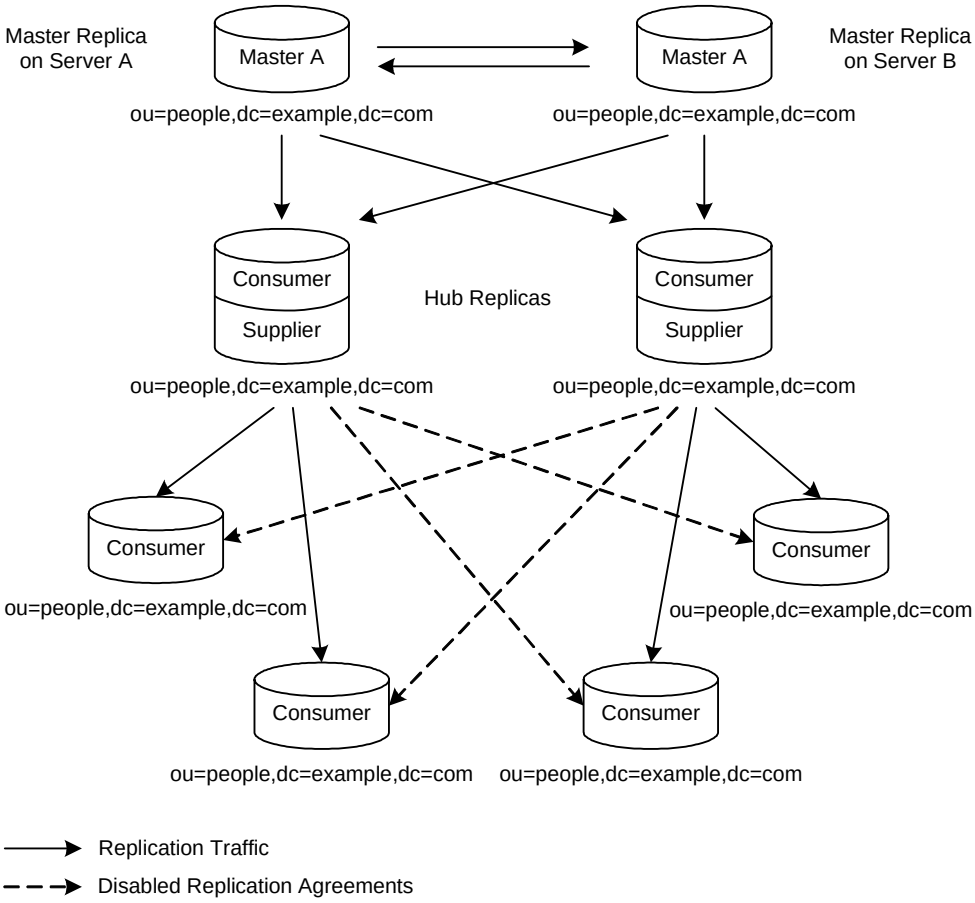
The consumer and the hub can process search requests received from clients, but in the case of modification requests, send the client a referral to the master. Figure 6-7 shows that Consumer C and D have a referral to Master A. These are the automatic referrals that are created when you create the replication agreement between the hub and the consumers. You can, however, as we have already stated, choose to overwrite these referrals should you wish to do so for performance or security reasons. For more information see Note on page 131.

NOTE You can combine multi-master and cascading replication scenarios. For example, in the multi-master scenario illustrated in Figure 6-8 on page 138, Server C and Server D could be hubs that would replicate to any number of consumers.

Mixed Environments

You can combine any of the scenarios outlined in the previous sections to best fit your needs. For example, you could combine a multi-master configuration with a cascading configuration to produce a topology similar to the scenario illustrated in Figure 6-8:

Figure 6-8 Combined Multi-Master and Cascading Replication



In the example illustrated in Figure 6-8, we have two masters and two hubs replicating data to four consumers. The hubs are used to balance the load of replication updates by sharing it between the masters and the hubs. This kind of configuration can prove to be valuable when you have a heavy load of replication updates to manage.

As in the example illustrated in Figure 6-7, both the hubs and the masters A and B, maintain change logs. It is, however, only the masters that can process directory modification requests from clients. When the hubs or the consumers receive modification requests from clients, they send the client a referral to the masters, in order for the request to be processed. The referrals are not indicated in Figure 6-8, but they exist between each of the four consumers and both masters, as well as between each of the hubs and the masters. These referrals are automatically created when you define your topology.

In the example illustrated in Figure 6-8 the dotted lines represent disabled replication agreements. If these replication agreements are not enabled, then the topology presented contains a single point of failure, if one of the hubs were to go off line. Whether or not you decide to enable the replication agreements to provide full read-write failover, will depend on your high availability requirements, but you need to be aware that by not enabling the agreements you *are* exposing yourself to a single point of failure risk.

Fractional Replication

In previous releases of Directory Server, the smallest unit of replication was the database and there was no way of replicating only a subset of the information inside a given database. Although the smallest unit of replication remains the database, Sun ONE Directory Server 5.2 offers new fractional replication functionality to cater for replication granularity requirements. This section is divided into two parts:

- What is Fractional Replication?
- Configuring Fractional Replication

What is Fractional Replication?

Fractional replication allows you to replicate a subset of the attributes of all entries in a given database from a supplier to a consumer. The following cases are just two examples of the scenarios where fractional replication can prove to be very useful:

- When you need to synchronize between intranet and extranet servers and filter out content for security reasons, fractional replication provides the filtering functionality.
- When you need to reduce replication costs, fractional replication allows you to be selective in what you choose to replicate. If your deployment only requires certain attributes to be available everywhere, then instead of replicating all attributes, you can use the fractional replication functionality to replicate required attributes only. For example you may want e-mail and phone attributes to be replicated but not all the other attributes that exist, particularly if the other attributes are ones that are modified quite frequently and as a result generate heavy traffic loads. Fractional replication allows you to filter in the required attributes and reduce traffic to a minimum. This filtering functionality can prove to be extremely valuable in replication environments where Directory Servers are separated by WANs.

CAUTION The fractional replication functionality provided in Sun ONE Directory Server 5.2 is *not* backward compatible with previous versions of Directory Server. If you are using fractional replication, you *must* ensure that *all* other instances of Directory Server are 5.2 instances.

Configuring Fractional Replication

In order to set up fractional replication you can either choose to exclude or include a list of attributes to be replicated, and this can be configured easily from the console. However, should you, at a later stage, wish to change your fractional replication configuration, you can do so as long as you remember to disable the replication agreements before proceeding to make any changes. Once you have made your changes you will need to enable your replication agreement again and re-initialize your consumers so that the new configuration is taken into account.

CAUTION There are two things to bear in mind when configuring fractional replication:

- When configuring fractional replication, it is essential that the server being replicated to be a read-only replica.
- We strongly recommend the use of an exclusion configuration approach. When we consider the complexity of certain features such as ACIs, CoS and Roles, and the dependency these features have on certain attributes, it becomes clear that managing a list of attributes to exclude is far safer, and less prone to human error, than managing a list of attributes to include.

Generally speaking you replicate all required attributes for each entry as defined in the schema, to avoid schema violations, but should you want to filter out some of the required attributes using the fractional replication functionality, then you need to be sure to disable schema checking. Having schema checking enabled with fractional replication can prevent you from being able to initialize off line, that is from an ldif file, because it would not allow you to load the ldif file if required attributes were filtered out. It is worth noting that turning schema checking off may have the added benefit of improving performance. It is also important to bear in mind that when 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 for different directory information trees on the same server instance.

Please note also that since 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 and, therefore, it will not correspond to the fractional replication configuration being applied.

Defining a Replication Strategy

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

- If high availability is your primary concern, you should create a data center with multiple directory servers on a single site. You can use single-master replication to provide read-failover or multi-master replication to provide write-failover. How to configure replication for high availability is described in “Using Replication for High Availability,” on page 146.
- If disaster recovery is your primary concern, you will want to create two distinct data centers, one in each geographical location, separated by WAN. Each data center will host two masters to provide failover and the fact that each data center is doubled, will protect you in the event of a disaster in one of the locations. To maintain write-failover high availability over geographically distributed sites, you can use four-way multi-master replication over a WAN.
- If local availability is your primary concern, you should use replication to geographically distribute data to directory servers in local offices around the world. You can decide to hold a master copy of all information in a single location, such as the company headquarters, or to let local sites manage the parts of the DIT that are relevant for them. The type of replication configuration to set up is described in “Using Replication for Local Availability,” on page 147.
- In all cases, you probably want to balance the load of requests serviced by your directory servers, and avoid network congestion. Strategies for load balancing your directory servers and your network are provided in “Using Replication for Load Balancing,” on page 148.

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

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

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

- Replication Backward Compatibility
- Replication Survey
- Replication Resource Requirements
- 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

Replication Backward Compatibility

One of the first things you need to establish is which versions of Directory Server you will be using in your replication configuration. In order to be sure that your replication configuration will function correctly we advise you take into account the information in Table 6-1 on page 143 which presents the possible master and consumer combinations between the different versions of Directory Server and their associated restrictions.

Table 6-1 Replication Backwards Compatibility Between 4.x, 5.0/5.1 and 5.2 Versions of Directory Server

	4.x Consumer	5.0/5.1 Consumer	5.0/5.1 Master	5.2 Consumer	5.2 Master	5.0/5.1/5.2 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

NOTE There are three important issues to bear in mind in terms of backwards compatibility:

- When 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 your 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.
- It is also important to understand that when you are replicating from a 5.2 server to a 5.0/5.1 the new 5.2 features and enhancements should not be used as they may result in the 5.0/5.1 servers behaving in unexpected ways.
- 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.

Replication Survey

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

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

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

For example, if your messaging server uses the directory, you need to know how many operations it performs for each 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 find out the type and frequency of operations that are performed in the directory.

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

The following sections will try to address these issues and guide you through the important issues you will need to consider when developing your replciation topology.

Replication Resource Requirements

Using replication requires more 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. For suppliers containing multiple replicated databases the change log will be used more frequently, and the disk usage will be even higher.

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

- Server threads.

  Each replication agreement creates two additional threads. The replication agreement threads are separate from the 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).

Using Replication for High Availability

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

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

NOTE You should not use replication as a replacement for a regular data backup policy. For information on backing up your directory data, refer to Backing Up Data section of the *Sun ONE Directory Server Administration Guide* and "Choosing a Backup Method," on page 265

If you need to guarantee write-failover for all your directory clients, you should use a multi-master replication scenario. The grouping and window mechanisms present in the multi-master replication flow allow you to configure your replication agreements in such a way as to optimize your replication performance. However, should read-failover be sufficient, you can use single-master replication.

LDAP client applications can usually be configured to search only one LDAP server. That is, unless you have written a custom client application to rotate through LDAP servers located at different DNS hostnames, you can only configure your LDAP client application to look at a single DNS hostname for a Directory Server. Therefore, you will probably need to use either DNS round robins or network sorts to provide fail-over to your backup Directory Servers. For information on setting up and using DNS round robins or network sorts, see your DNS documentation.

With regard to maintaining write-failover high availability over two geographically distributed sites, you can use four-way multi-master replication over a WAN. You set up two master servers in one location and two master servers in the second location and configure them to be fully-connected over a WAN, to safeguard against the eventuality of one master going off line. As with multi-master replication over a LAN, you can use the grouping and window mechanisms to optimize your replication performance.

Alternatively, you can use the Sun ONE Directory Proxy Server product. For more information on Sun ONE Directory Proxy Server, go to `http://www.sun.com/software`.

Using Replication for Local Availability

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

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

- You need a local master copy of the data.

  This is an important strategy for large, multinational enterprises that need to maintain directory information of interest only to the employees in a specific country. Having a local master copy of the data is also important to any enterprise where interoffice politics dictate that data be controlled at a divisional or organizational level.

- You are using unreliable or intermittently available network connections.

  Intermittent network connections can occur if you are using unreliable WANs, such as often occurs in international networks.

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

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

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

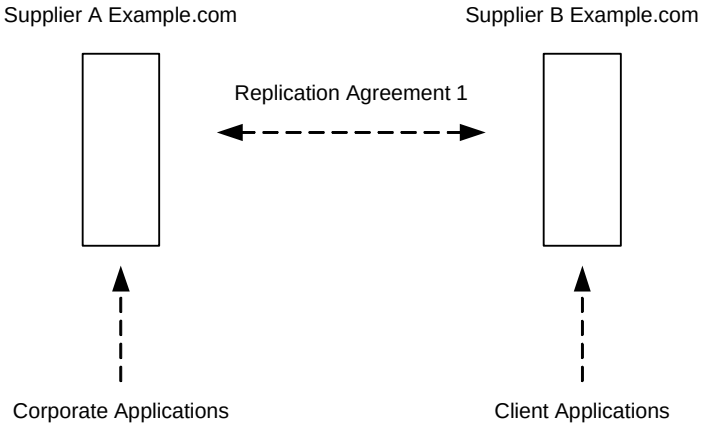
  Your network may be perfectly reliable and available, but you nevertheless want to reduce the cost on your network.

Using Replication for Load Balancing

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

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

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



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

Directory entries generally average around one KB in size. Therefore, 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. Given a slow, heavily loaded, or unreliable WAN, you may need to replicate your directory tree to a local server.

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

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

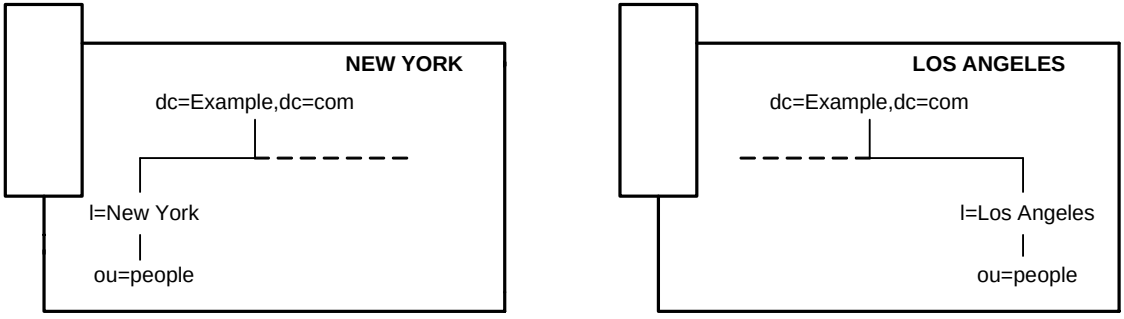
Given the difference in loads caused by replication versus that caused by normal directory usage, you may decide that replication for network load-balancing purposes is not desirable. On the other hand, you may find that the benefits of locally available directory data far outweigh any considerations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

A good compromise between making data available to local sites without overloading the network is to use scheduled replication. For more information on data consistency and replication schedules, refer to "Data Consistency," on page 124.

Example of Network Load Balancing

Suppose your enterprise has offices in two cities. Each office has specific subtrees that they manage, as illustrated in Figure 6-10:

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



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

- Select one server in each office to be the master 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 allows for optimized performance.

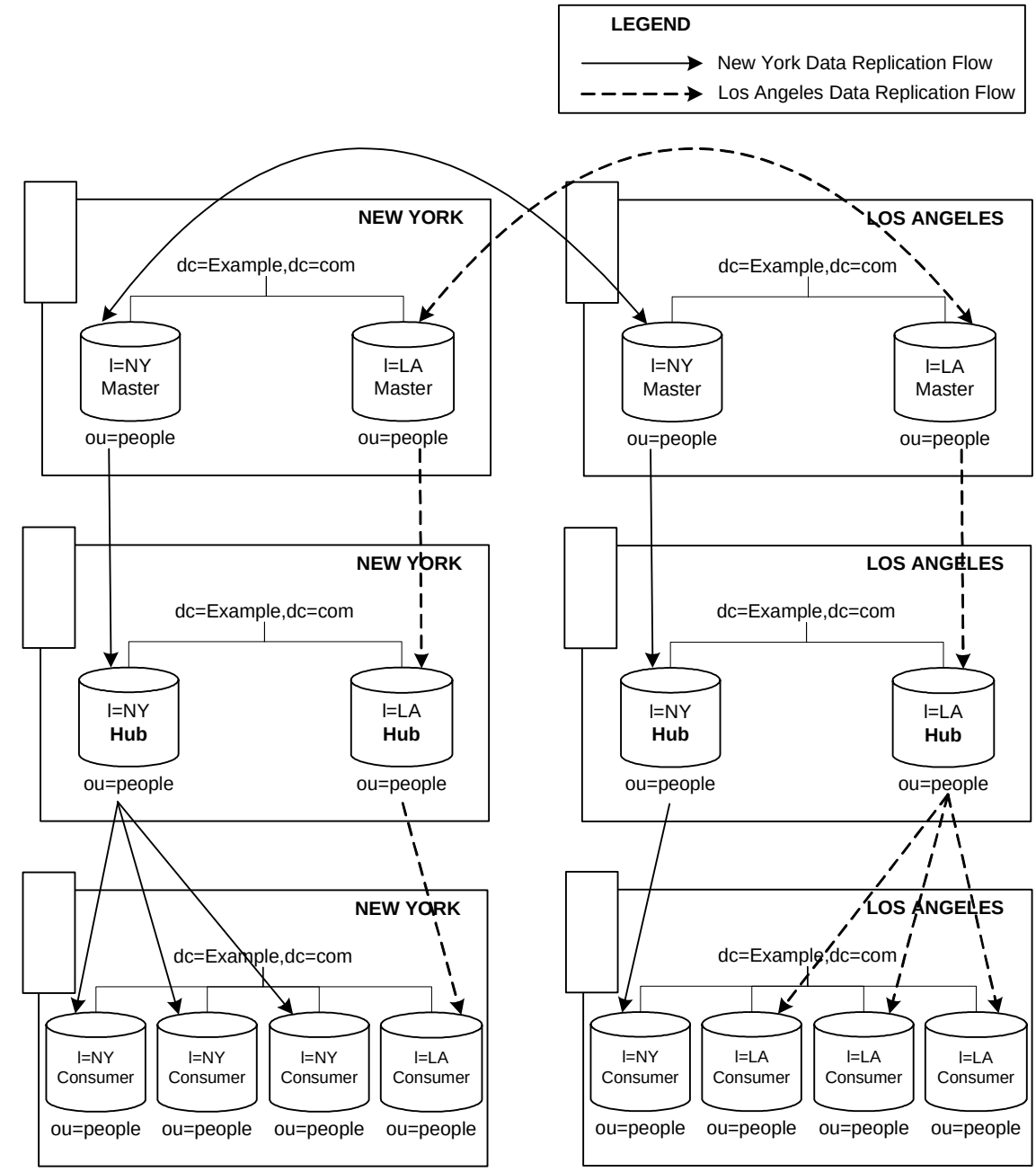
- Replicate the directory tree on each master (including data supplied from the remote office) to at least one local Directory Server to ensure availability of the directory data.

- 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 LA 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 on page 151:

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



Example of Load Balancing for Improved Performance

Suppose that your directory must include 15,000,000 entries in support of 10,000,000 users, and that each user performs ten directory lookups a day. Also assume that you are using a messaging server that handles 250,000,000 mail messages a day, and that performs five directory lookups for every mail message that it handles. So, you can expect 1,250,000,000 directory lookups per day just as a result of mail. Your total combined traffic is, therefore, 1,350,000,000 directory lookups per day.

Assuming an eight-hour business day, and that your 10,000,000 directory users are clustered in four time zones, your business day (or peak usage) across four time zones is 12 hours long. So, you must support 1,350,000,000 directory 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

Now, assume that you are using a combination of CPU and RAM with your Directory Servers that allows you to support 5,000 reads per second. Simple division indicates that you need at least six or seven Directory Servers to support this load. However, for enterprises with 10,000,000 directory users, you should add more Directory Servers for local availability purposes.

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

You could, therefore, replicate as follows:

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

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

- Use these 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 a definition of a hub, refer to “Cascading Replication,” on page 134.

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

  Replicating to local sites helps balance the work load of your servers and your WANs, as well as ensuring high availability of directory data. Assume that you want to replicate to four sites around the country. You then have four consumers for each hub.

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

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

Example Replication Strategy for a Small Site

Suppose your entire enterprise is contained within a single building. This building has a very fast (100 MB per second) and lightly used network. The network is very stable and you are reasonably confident of the reliability of your server hardware and OS platforms. 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 in the event that 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 ONE Directory Proxy Server. For more information on Sun ONE Directory Proxy Server, go to `http://www.sun.com/software`.

Example Replication Strategy for a Large Site

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

Your replication strategy follows:

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

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

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

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

- Create two replicas in the second building (Building B).

- If there is no need for close consistency between the 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 sites - one in France and the other site 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 connections are very busy during normal business hours.

Your replication strategy follows:

- Hold master copies of your directory data on servers in both geographical locations.

- For write-failover within your French and American sites, replicate your data to a second master located within each geographical location.
- Deploy a fully-connected, four-way, multi-master replication topology between France and the USA to provide complete high-availability and write-failover cover across your enterprise deployment.
- Deploy as many consumers as you require in each geographical location to reduce the load on your masters as far as possible in terms of 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 help optimize your bandwidth capabilities.

Using Replication with Other Directory Features

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

Replication and Access Control

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

For more information about designing access control for your directory, refer to Chapter 7, “Designing a Secure Directory,” on page 163.

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 listed 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 is supported to provide backward compatibility with 4.x releases of Directory Server and was not designed to function in a multi-master replication environment. The Retro Change Log Plug-in stores changes in the order of arrival on the local server and *not* in the order in which these changes were applied to the system. If the Retro Change Log Plug-in is configured on 2 servers there is no guarantee that the changes will be logged in the same order with the same sequence numbers. As the order of changes is fundamental to the replication process, this means that when you are using the Retro Change Log Plug-in in a multi-master replication context, you can use it to see what changes have actually been logged, but you should not rely on the actual content of the changes applied.

Replication and the Referential Integrity Plug-In

You can use the referential integrity plug-in with multi-master replication provided that this plug-in is enabled on all master replicas.

NOTE By default the referential integrity plug-ins is disabled, so you need to remember to enable it using the 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.

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

When pre- and post-operation plug-ins are used in a replication context, replication must be able to detect the order of these pre- and post-operation plug-ins. You can decide whether or not to make changes to these replication operations, but it is worth noting that if operations are replicated operations, then changing them can result in unexpected behavior. For more information on pre- and post-operation plug-ins, refer to “Extending Client Request Handling” in the *Sun ONE Directory Server Plug-In API Programming 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, or the farm server. In this environment, you cannot replicate the chained suffix itself. You can, however, replicate the database that contains the actual data on the remote server.

NOTE You must configure the replication agreement on the farm server and not on the multiplexor.

You must not use the replication process as a backup for chained suffixes. You must back up chained suffixes manually. For more information about chaining and entry distribution, refer to Chapter 5,

Schema Replication

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

The best way to guarantee schema consistency is to make schema modifications on a single master server, even in the case of a multi-master replication environment.

Schema replication happens automatically. If replication has been configured between a supplier and a consumer, schema replication will happen by default.

NOTE Directory Server 5.2 offers a new attribute called the `nsslapd-schema-repl-useronly` attribute which can be set so that only user defined schema is replicated, that is, only the schema which is added over LDAP or added as files with the `'user defined'` value in the `X-ORIGIN` field. This makes it possible to reduce the amount of data transferred and thus speed up the replication of schema.

The logic used by Directory Server for schema replication is the same in every replication scenario, and can be described as follows:

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

   It is interesting to note that Directory Server 5.2 has an attribute which allows you to replicate only user-defined schema, that is schema which has been added over LDAP or added as a file with an `X-ORIGIN` value of `'user defined'`. This allows you to reduce the amount of data being transferred should you so desire, and speed up the schema replication process.

NOTE Note that in contrast to previous versions of Directory Server, ACIs present in the schema are now replicated.

If you make schema modifications on two master servers in a multi-master set, whichever master was updated last will "win" and its schema will be propagated to the consumer. This means that you risk losing the modifications you make to one master, if different modifications are made to the other master at a later stage. To avoid losing modifications, always make sure you make schema modifications on one master only.

NOTE You must *never* update the schema on a consumer because the supplier is unable to resolve the conflicts that will occur and replication will fail. If you do 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 if you search on a consumer or try to perform an update operation on a supplier.

Schema should be maintained on a single master in a multi-master replicated topology. If you are using the standard `99user.ldif` file, these changes will be replicated to all consumers. When you are using custom schema files, ensure that these files are copied to all servers after making changes on the master. After copying files, the server must be restarted. Refer to "Creating Custom Schema Files - Best Practices and Pitfalls," on page 49 for more information.

Changes made to custom schema files are only replicated if the schema is updated using LDAP or the Directory Server Console. These custom schema files should be copied to each server in order to maintain the information in the same schema file on all servers. For more information, refer to “Creating Custom Schema Files - Best Practices and Pitfalls,” on page 49.

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

Replication and Multiple Password Policies

In an environment that uses multiple password policies, you need to be sure to replicate the LDAP subentry that contains the definition of the policy to apply to the replicated entries. If you fail to do so, the default password policy will be applied and will of course not work for entries that have been configured to use a non-default password policy. It is important to understand that if you replicate these entries to a 5.0/5.1 server, the replication will function correctly, but the password policy will not be enforced on the 5.0/5.1 server as the possibility of having multiple password policies is specific to Directory Server 5.2.

Replication Monitoring

Sun ONE Directory Server 5.2 provides replication monitoring tools that allow you to monitor replication between servers. Being able to monitor replication activity assists in identifying the causes of replication problems and troubleshooting. All of the Directory Server replication monitoring tools can be used when LDAPS is turned on. The three replication monitoring tools are:

- insync
- entrycmp
- repldisc

For more information regarding these replication monitoring tools, refer to the Replication Monitoring Tools section of the *Sun ONE Directory Server Reference Manual* and for more information on the monitoring possibilities afforded to you by certain replication attributes, see the replication attributes in the Core Server Configuration Attributes chapter of the *Sun ONE Directory Server Reference Manual*.

NOTE It is important to understand that these tools constitute an LDAP client, and as such, will need to authenticate to the server and use a bind DN that has read access to `cn=config`.

insync

The `insync` tool indicates the state of synchronization between a master replica and one or more consumer replicas. Being aware of the degree of synchronization is vital when it comes to managing potential conflicts.

entrycmp

The `entrycmp` tool allows you to compare the same entry on two or more 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. All of the entries' attributes and values are compared and if everything is identical, the entries are considered to be the same.

NOTE If the machine on which you are running either `insync` or `entrycmp` cannot reach the host about which it is inquiring, whether this be due to a firewall, VPN, or other network setup reasons, for example in a topology with a hub, you may encounter difficulties using the `insync` and `entrycmp` tools.

repldisc

The `repldisc` tool allows you to discover a replication topology. Topology discovery starts with one server and builds 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.

NOTE When using the replication monitoring tools, it is important to bear two things in mind:

- First, you must be sure to use either all symbolic names or all IP addresses when identifying hosts. Using a combination of the two can be problematic.
- Second, when SSL is enabled, the directory on which you are running the tools must have a copy of all the certificates used by the other servers in the topology.

Chapter 7

Designing a Secure Directory

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

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

- About Security Threats
- Analyzing Your Security Needs
- Overview of Security Methods
- Selecting Appropriate Authentication Methods
- Preventing Authentication by Account Inactivation
- Designing your Password Policies
- Designing Access Control
- Securing Connections With SSL
- Encrypting Attributes
- Grouping Entries Securely
- Securing Configuration Information
- Other Security Resources

About Security Threats

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

- Unauthorized Access
- Unauthorized Tampering
- Denial of Service

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

Unauthorized Access

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

For example, an unauthorized client can use another client's credentials to access the data. Or an unauthorized client can eavesdrop on the information exchanged between a legitimate client and Directory Server.

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

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

The authentication methods, password policies, and access control mechanisms provided by the Sun ONE Directory Server offer efficient ways of preventing unauthorized access. Refer to "Selecting Appropriate Authentication Methods," on page 169, "Designing your Password Policies," on page 175, and "Designing Access Control," on page 189, for more information about these topics.

Unauthorized Tampering

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

For example, if your directory cannot detect tampering, an attacker could change a client's request to the server (or not forward it) and change the server's response to the client. SSL and similar technologies can solve this problem by signing information at either end of the connection. For more information about using SSL with Sun ONE Directory Server, refer to "Securing Connections With SSL," on page 209.

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.

Sun ONE Directory Server offers a way of preventing denial of service attacks by setting limits on the resources allocated to a particular bind DN. For more information about setting resource limits based on the user's bind DN, refer to the Setting Resource Limits Based on the Bind DN section of the *Sun ONE Directory Server Administration Guide*.

Analyzing Your Security Needs

You need to analyze your environment and users to determine your specific security needs. When you performed your site survey in Chapter 2, "Planning and Accessing Directory Data" you made some basic decisions about who can read and write the individual pieces of data in your directory. This information now forms the basis of your security design.

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

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

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

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

- To offer your customers and business partners a guarantee of privacy.
- To guarantee information integrity.

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

- Determining Access Rights
- Ensuring Data Privacy and Integrity
- Conducting Regular Audits
- Example Security Needs Analysis

Determining Access Rights

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

You may 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.

No matter 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 169. For information about restricting access to directory information, refer to “Designing Access Control,” on page 189.

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 the Sun ONE Directory Server, refer to “Password Storage Scheme,” on page 179 and “Encrypting Attributes,” on page 211. For information about signing data, refer to “Securing Connections With SSL,” on page 209.

Conducting Regular Audits

As an extra security measure, you should conduct regular audits to verify the efficiency of your overall security policy. You can do this by examining the log files and the information recorded by the SNMP agents. For more information about monitoring your directory, refer to Chapter 8, “Monitoring Your Directory.”

Example Security Needs Analysis

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

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

Therefore, Example.com has three main categories of information in its directory:

- Example.com internal information

- Information belonging to corporate customers
- Information pertaining to individual subscribers

Example.com needs the following access controls:

- Provide access to the directory administrators of Example2 and Example3 to their own directory information.
- Implement Example2's and Example3's own access control policies for their directory information.
- Implement a standard access control policy for all individual clients who use Example.com for Internet access from their homes.
- Deny access to Example.com's corporate directory to all outsiders.
- Grant read access to Example.com's directory of subscribers to the world.

Overview of Security Methods

Sun ONE Directory Server offers a number of methods that you can use to design an overall security policy that is adapted to your needs. Your security policy should be strong enough to prevent sensitive information from being modified or retrieved by unauthorized users while simple enough to administer easily. A complex security policy can lead to mistakes that either prevent people from accessing information that you want them to access or, worse, allow people to modify or retrieve directory information that you do not want them to access.

Sun ONE 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.

Selecting Appropriate Authentication Methods

A basic decision you need to 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?

Sun ONE 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

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

For information about preventing authentication by a client or group of clients, see “Preventing Authentication by Account Inactivation,” on page 174.

Anonymous Access

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

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

Therefore, if you attempt to block a specific user or group of users from seeing some kinds of directory data, but you have allowed anonymous access to that data, then those users can still access the data simply by binding to the directory anonymously.

You can restrict the privileges of anonymous access. Usually directory administrators only allow anonymous access for read, search, and compare privileges (not for write, add, delete, or selfwrite). Often, administrators limit access to a subset of attributes that contain general information such as names, telephone numbers, and email addresses. Anonymous access should never be allowed for more sensitive data such as government identification numbers (social security numbers in the US), home telephone numbers and addresses, and salary information.

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

- 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 "dc=Example,dc=com cn=joe" objectclass=*
```

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, you must authenticate to Directory Server before you can access the directory contents. With simple password authentication, a client authenticates to the server by sending a simple, reusable password.

For example, a client authenticates to 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 over the net. 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

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

For example, 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.

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

For more information on proxy authorization, refer to Chapter 6, “Managing Access Control” in the *Sun ONE 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 209. For information about configuring certificates and activating SSL, see Chapter 11, "Implementing Security" in the *Sun ONE 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 10 of the *Managing Servers with Sun ONE Console*. See also "Configuring Certificate-Based Authentication in Clients" in Chapter 11 of the *Sun ONE Directory Server Administration Guide*.

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 Environment, the General Security Services API (GSSAPI) allows Directory Server to interact with the Kerberos V5 security system to positively identify a user. The client application must present its credentials to the Kerberos systems, 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 "SASL Authentication Through DIGEST-MD5" and "SASL Authentication Through GSSAPI (Solaris Only)" in Chapter 11 of the *Sun ONE 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.

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 72.

Designing your Password Policies

A password policy is a set of rules that govern how passwords are administered in a given system. Password policy in Directory Server defines the following:

- password change policy
- password minimum length
- password maximum age
- password expiration policy and its associated warning procedure
- password syntax check policy
- password storage scheme used
- password history procedure
- password failure record procedure
- account lockout procedure

In contrast to previous releases of Directory Server, password policy functionality provided by Sun ONE Directory Server 5.2 offers increased flexibility in that you can configure multiple password policies as opposed to one global policy for your entire directory. You can configure multiple password policies and you can choose to assign them either to particular users or to whole sets of users using the CoS and Roles functionality. This affords users and administrators of Directory Server significantly more scope when it comes to implementing password policy security measures, because they can tailor password policies to specific users or roles and thus cater precisely for their complex security requirements.

This section will begin with a presentation of basic password policy features. We will then take a look at the different ways in which you can configure your password policy, and examine the order of precedence that governs the application of multiple password policies. Finally, we will examine account lockout policy and the implications of designing password policies in a replicated environment. For detailed information regarding the attributes users have at their disposal to build a password policy that is suited to their needs, see the Password Policy Attributes and Account Lockout Attributes sections in the *Sun ONE Directory Server Reference Manual*. This section is divided into the following parts:

- Password Policy Features
- Configuring Your Password Policies
- Designing an Account Lockout Policy

- Designing Password Policies in a Replicated Environment

Password Policy Features

This section takes you through the main password policy features and is divided into the following sub-sections:

- User-Defined Passwords
- Password Change After 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 After First Login or Reset

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

Often the initial passwords set by the administrator follow some sort of convention, such as the user's initials, user ID, or the company name. Once the convention is discovered, it is usually the first value tried by a hacker trying to break in. In this case, it is a good idea to require users to change their passwords after such a change. If you configure this option for your password policy, users are required to change their password even if user-defined passwords are disabled. For further information see "User-Defined Passwords," on page 176.

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

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

Password Expiration

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

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 decide to re-enable password expiration, the default password expiration duration is 90 days because that is what you had it set to before you disabled the feature.

By default, user passwords never expire.

Expiration Warning

If you choose to set your password policy so that user passwords expire after a given number of days, it is a good idea to send users a warning before their passwords expire. You can set your policy so that users are sent a warning 1 to 24,855 days before their passwords expire. Directory Server displays the warning when the user binds to the server. If password expiration is turned on, by default, a warning is sent (via 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 the password syntax guidelines established by the password policy. By default, password syntax checking is turned off.

Password Length

Directory Server allows you to specify a minimum length for user passwords. In general, shorter passwords are easier to crack. You can require passwords that are from 2 to 512 characters. A good length for passwords is 8 characters. This is long enough to be difficult to crack, but short enough 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 Directory Server to not allow users to change their passwords for a given time. You can use this feature in conjunction with the `passwordInHistory` attribute to discourage users from reusing old passwords.

Setting the password minimum age (`passwordMinAge`) attribute 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 by entering an integer value in the `passwordInHistory` attribute. If you enter a value of `0`, then the password history function will not be enabled. As a result, no previously used passwords will be stored and users will be able to reuse passwords.

If, however, you enter a value between `1` and `24` in the `passwordInHistory` attribute, the directory will store that number of old passwords in the `passwordHistory` attribute. If a user attempts to reuse one of the passwords Directory Server has stored, the directory rejects the password. This feature prevents users from reusing one or two passwords that are easy to remember.

By default, Directory Server does not store any previously used passwords.

Password Storage Scheme

The password storage scheme specifies the type of encryption used to store Directory Server passwords within the directory. You can specify:

- Clear text (no encryption).
- Secure Hash Algorithm (SHA).
- Salted Secure Hash Algorithm (SSHA). This encryption method is the default 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 of the choices and is the default hash algorithm for Directory Server.

Configuring Your Password Policies

In Sun ONE Directory Server 5.2 you have four password policy options available to you. Directory Server provides you with a default password policy, which will be applied automatically, the parameters of which you can change should you wish to do so. As a backup to the default password policy, Directory Server also provides a hard-coded password policy, which is applied should the default password policy be absent or, following modifications, no longer valid. The

attribute values of the hard-coded password policy are the same as the default password policy values. Otherwise you can choose to define a password policy and apply it to a particular user, or to a set of users using the CoS and Roles functionality.

This section will describe each of these password policy options in more detail, and will explain 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 parts:

- Default Password Policy
- Defining Password Policies for Users or Sets of Users
- Multiple Password Policies and Their Order of Precedence

Default Password Policy

The default password policy provided with Sun ONE Directory Server is, as its name suggests, the password policy that is used should users not design their own.

NOTE Should you wish to change any of the default password policy attribute values, do not forget that in contrast to previous releases of Directory Server, the password policy attributes are no longer stored directly under `cn=config`, but instead under `cn=Password Policy,cn=config`. If this entry does not exist then the hard-coded password policy provided with Directory Server will be applied.

For more information about the password policy attributes see the Password Policy Attributes section in the *Sun ONE Directory Server Reference Manual.*

By default your password policy will enforce the following:

- The SSHA storage scheme.
- Users can change their passwords.
- Users do not have to change their password after their first login or after the password is reset by the administrator.
- Password syntax checks (i.e., compliance with the minimum number of characters) will not be performed.
- Passwords will never expire.

- The maximum age of a password is 100 days but only if you decide to activate password expiration.
- No time needs to elapse between modifications to the password.
- If you decide to activate the password expiration mechanism, a password expiration warning will be sent 1 day before the password is due to expire on your first bind attempt.
- Passwords used will not be recorded.
- Users are never locked out of their accounts.
- If you decide to activate the account lockout mechanism, then by default users will be locked out after a maximum number of 3 failed bind attempts, and the lockout will last for 1 hour.
- Password failures are purged from the failure counter after 600 seconds.

How you adapt the default password policy will of course depend on how stringent your security requirements are. A password policy such as the default password policy where passwords never expire, where no syntax checks are performed, and which does not have an account lockout mechanism enabled does not come without its security risks. You will have to be sure to balance your security requirements against the management overheads generated by a demanding password policy. Once you have established your precise security requirements and deployed a certain password policy solution, you will be able to analyze its pros and cons, and make changes should you be concerned that your chosen policy does not provide sufficient security or is proving to be too unwieldy.

Defining Password Policies for Users or Sets of Users

To define a password policy for a given user entry or a set of users a new attribute called `passwordPolicySubentry` is used. 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, in the sense that you enter it, or a virtual attribute, in that it is generated by a CoS definition. A natural way of defining password policies for a set of users is to configure the CoS definition to provide values for the `passwordPolicySubentry` attribute in user entries as a function of the Roles that those user entries have. Assigning password policies to sets of users by assigning them as a function of the Roles those users have is not the only possibility you have available to you, but it is the one presented to you by Directory Server Console.

Defining a Password Policy for a User

When you want to configure a password policy for a given user you must define it for a particular subtree by adding an LDAP entry whose immediate superior is the root of the subtree in question.

CAUTION Users must *not* be able to modify their own `passwordPolicySubentry` attribute which should be controlled by an ACI.

Imagine that as system administrator of the company Example.com, you want to apply a more stringent password policy called `strictPwdPolicy` to a user in the `dc=example,dc=com` subtree. The `strictPwdPolicy` password policy you wish to apply, in contrast to the default policy, performs syntax checks, enforces the expiration of passwords after 10 days, and proceeds with account lockout once 3 consecutive bind attempts have failed. You would add the following `passwordPolicySubentry` attribute directly to the user entry in the `dc=example,dc=com` subtree:

```
passwordPolicySubentry:cn=strictPwdPolicy,dc=example,dc=com
```

The value of the above `passwordPolicySubentry` attribute is the DN of the LDAPsubentry that stores attributes which define the `strictPwdPolicy` password policy. This LDAPsubentry for your `strictPwdPolicy` password policy would read as follows:

```
dn:cn=strictPwdPolicy,dc=example,dc=com
objectclass:top
objectclass:passwordPolicy
objectclass:LDAPsubentry
passwordStorageScheme:SSHA
passwordChange:on
passwordMustChange:on
passwordCheckSyntax:on
passwordExp:on
passwordMinLength:6
passwordMaxAge:8640000
passwordMinAge:0
passwordWarning:8640000
passwordInHistory:6
passwordLockout:on
passwordMaxFailure:3
passwordUnlock:off
passwordLockoutDuration:3600
passwordResetFailureCount:600
```

Defining a Password Policy for a Set of Users Using CoS and Roles Functionality

In the same way, imagine that as system administrator of the company Example.com, where contractors are assigned to a `contractor` managed role, and all in-house employees are assigned to an `employee` managed role, you decide that you want to apply your more stringent password policy `veryStrictPwdPolicy` to all contractors and a less stringent password policy called `normalPwdPolicy` to your employees.

Table 7-1 on page 184 presents your `contractor` role and `employee` role definitions:

Table 7-1 Contractor and Employee Role Definitions

Contractor Role Definition	Employee Role Definition
dn:cn=contractorRole,dc=example,dc=com objectclass:LDAPSubentry objectclass:nsRoleDefinition objectclass:nsSimpleRoleDefinition objectclass:nsManagedRoleDefinition cn:contractorRole description:managed role for contractors	dn:cn=employeeRole,dc=example,dc=com objectclass:LDAPSubentry objectclass:nsRoleDefinition objectclass:nsSimpleRoleDefinition objectclass:nsManagedRoleDefinition cn:employeeRole description:managed role for in-house employees

By defining a CoS definition entry and a CoS template entry that contain the required password policy for both roles, your `passwordPolicySubentry` attributes (which point to DN of the LDAPsubentry containing the appropriate password policy) will be generated for each role.

NOTE When assigning a password policy to (members of) a role, do NOT add the passwordPolicySubentry attribute to the role itself , but instead to the CoS associated with that role.

You will want to create one CoS definition entry that will generate the `passwordPolicySubentry` attribute values for each role. The CoS definition entry you will need to create is presented in Table 7-2 on page 184:

Table 7-2 CoS Definition Entry for Password Policy

CoS Definition Entry
dn:cn=PwdPol_cosDefinition,dc=example,dc=com objectclass:top objectclass:LDAPsubentry objectclass:cosSuperDefinition objectclass:cosClassicDefinition cosTemplateDn:cn=PwdPolTemplContainer,dc=example,dc=com cosSpecifier:nsRole cosAttribute:passwordPolicySubentry operational

By specifying the following `operational` qualifier:

```
cosAttribute:passwordPolicySubentry operational
```

you are specifying that you want the generated, and therefore virtual, value of the `cosAttribute` to override any real attribute that may exist. It is in actual fact necessary to specify the `operational` qualifier here. For more information on Roles and CoS, see Chapter 4, “Designing the Directory Tree.”

Once you have created the CoS definition entry, you will then create the CoS template entries that contain the values of the virtual `passwordPolicySubentry` attributes. In our example, the virtual value of the `passwordPolicySubentry` attribute for the `contractor` managed role will be the `dn` of the `strictPwdPolicy` LDAPsubentry, and the `dn` of the `normalPwdPolicy` LDAPsubentry for the `employee` managed role. The CoS template entry you will need to create for the `contractor` managed role is presented below in Table 7-3 on page 185:

Table 7-3 CoS Template Entry for Contractor Role

CoS Template Entry for Contractor Role

`dn:cn=\"cn=ContractorRole,dc=example,dc=com\",`
` cn=PwdPolTemplContainer,dc=example,dc=com`
`objectclass:top`
`objectclass:extensibleObject`
`objectclass:costemplate`
`objectclass:ldapsubentry`
`cosPriority:1`
`passwordPolicySubentry:cn=veryStrictPwdPolicy,dc=example,dc=com`

and the CoS template entry you will need to create for the `employee` managed role is presented below in Table 7-4 on page 185

Table 7-4 CoS Template Entry for Employee Role

CoS Template Entry for Employee Role

`dn:cn=\"cn=EmployeeRole,dc=example,dc=com\",`
`cn=PwdPolTemplContainer,dc=example,dc=com`
`objectclass:top`
`objectclass:extensibleObject`
`objectclass:costemplate`
`objectclass:ldapsubentry`
`cosPriority:1`
`passwordPolicySubentry:cn:cn=normalPwdPolicy,dc=example,dc=com`

The above entries also show that you have decided to store your template entries in a container called `cn=PwdPolTempContainer,dc=example,dc=com` which appears in ldif as follows:

```
dn:cn=PwdPolTempContainer,dc=example,dc=com
objectclass:top
objectclass:nsContainer
```

NOTE For easier password policy management, we advise you wherever possible to co-locate the user or set of users to which your password policy applies and the password policy itself. This will help you guard against forgetting to replicate the password policy LDAP subentry.

Multiple Password Policies and Their Order of Precedence

Now that Directory Server allows for multiple password policies, we can easily imagine the scenario where a password policy exists for a user entry while that same user entry belongs to a role that also has a password policy assigned to it. Which password policy takes precedence? In order to deploy password policies that actually correspond to your security needs, you need to understand the order of precedence that governs their application and how to control that order when defining your CoS template entries.

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 same user entry. This is true 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, “Designing the Directory Tree.”
2. A password policy assigned to a user entry will take precedence over the default password policy.

3. The default 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 the event that a user entry is affected by more than one CoS generated password policy, You specify the desired 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. Again, for more information on Roles and CoS, see Chapter 4, "Designing the Directory Tree."

Designing an Account Lockout Policy

Once you have established the password policies for your directory, you can protect your user passwords from potential threats by configuring an account lockout policy.

The lockout policy works in conjunction with the password policy to provide further security. You can set up your password policy so that a specific user is locked out of the directory after a given number of failed attempts to bind.

The account lockout feature protects against hackers who try to break into the directory by repeatedly trying to guess a user's password. 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.

Designing Password Policies in a Replicated Environment

Password and account lockout policies are enforced in a replicated environment as follows:

- Password policies are enforced on the master copy of the data.
- Account lockout is enforced on all servers participating in replication.

Some of the password policy state information in your directory is replicated. The replicated attributes are:

- `passwordHistory`
- `passwordAllowChangeTime`
- `passwordExpirationTime`

CAUTION However, the configuration information stored under `cn=Password Policy,cn=config` is kept locally and is *not* replicated. This information includes the password syntax and the history of password modifications. Account lockout counters are *not* replicated either.

When configuring password policies in a replicated environment, consider the following points:

- In an environment that uses multiple password policies, you need to be sure to replicate the LDAP subentry that contains the definition of the policy to apply to the replicated entries. If you fail to do so, the LDAP subentry containing the definition of your policy will not exist and the default password policy will be applied.
- All replicas issue warnings of an impending password expiration. This information is kept locally on each server, so if a user binds to several replicas in turn, the user receives the same warning several times. In addition, if the user changes the password, it may take time for this information to be updated on the consumer replicas. If a user changes a password and then immediately rebinds, the bind may fail until the consumer replica registers the changes made to the master replica.
- You want the same bind behavior to occur on all servers, including suppliers and consumers. Make sure you create the same password policy configuration information on each server.
- Account lockout counters may not work as expected in a multi-master environment, given that they are not replicated.
- Entries that are created for replication (for example, the server identity Replication Manager entries) need to 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` (i.e. the maximum value in the valid range).

Designing Access Control

Once you decide on one or more authentication schemes to establish the identity of directory clients, you need to decide how to use the schemes to protect information contained in your directory. Access control allows you to specify that certain clients have access to particular information, while other clients do not.

You specify access control using one or more access control lists (ACL). Your directory's ACLs consist of a series of one or more access control information (ACI) statements that either allow or deny permissions (such as read, write, search, proxy, add, delete, and compare) to specified entries and their attributes.

Using the ACL, you can set permissions for the following:

- The entire directory
- A particular subtree of the directory
- Specific entries in the directory
- A specific set of entry attributes
- Any entry that matches a given LDAP search filter

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

This section will examine the Sun ONE Directory Server access control mechanism and is divided into the following parts:

- About the ACI Format
- Default ACIs
- Deciding How to Set Permissions
- Requesting Effective Rights Information
- Tips on Using ACIs
- ACI Limitations

About the ACI Format

When designing your security policy, it is helpful to understand how ACIs are represented in Directory Server. It is also helpful to understand what permissions you can set in your directory. This section gives you a brief overview of the ACI mechanism. For a complete description of the ACI format, see the Managing Access Control chapter of the *Sun ONE Directory Server Administration Guide*.

Access control instructions 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. Directory ACIs take the following general form:

target permission bind_rule

The ACI variables are defined below:

- *target*

  Specifies the entry (usually a subtree) that the ACI targets, the attribute it targets, or both. In other words, the *target* identifies the directory element that the ACI applies to. An ACI can target only one entry, but it can target multiple attributes. In addition, the *target* can contain an LDAP search filter. This allows you to set permissions for widely scattered entries that contain common attribute values.

- *permission*

  Identifies the actual permission being set by this ACI. The *permission* says that the ACI allows or denies a specific type of directory access, such as read, write, search, proxy, add, delete, and compare to the specified *target*.

- *bind_rule*

  Identifies the bind DN or network location to which the permission applies. The bind rule may also specify an LDAP filter, and if that filter is evaluated to be true for the binding client application, then the ACI applies to the client application.

So, ACIs are expressed as follows:

"For the directory object *target*, allow or deny *permission* if the *bind_rule* is true."

An example ACI which allows all users search, read and compare permissions for all attributes would appear as follows:

```
aci: (targetattr = "*")(version 3.0; acl "my aci"; allow
 (search,read,compare) userdn="ldap:///all";)
```

The *permission* and *bind_rule* portions of the ACI are set as a pair, and are also called an Access Control Rule. You can have multiple *permission bind_rule* pairs for every target. This allows you to efficiently set multiple access controls for any given target. For example:

target (permission bind_rule) (permissions bind_rule)...

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

Targets

You must decide what entry is targeted by every ACI you create in your directory. If you target a directory entry that is a directory branch point, then that branch point, as well as all of its child entries, is included in the scope of the permission. The advantage of this is that you can place at a high level in the directory tree a general ACI that effectively applies to entries more likely to be located lower in the tree. For example, at the level of an `organizationalUnit` entry or a `locality` entry, you could create an ACI that targets entries that include the `inetorgperson` object class. You can use this feature to minimize the number of ACIs in the directory tree by placing general rules at high level branch points. To limit the scope of more specific rules, you should place them as close as possible to leaf entries.

If you do not explicitly specify a target entry for the ACI, then the ACI is targeted to the directory entry that contains the ACI statement. Also, the default set of attributes targeted by the ACI is any attribute available in the targeted entry's object class structure.

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

NOTE ACIs placed in the root DSE entry apply only to that entry.

In addition to targeting entries, you can also target attributes on the entry. This allows you to set a permission that applies to only a subset of attribute values. You can target sets of attributes by explicitly naming those attributes that are targeted, or by explicitly naming the attributes that are not targeted by the ACI. Use the latter case if you want to set a permission for all but a few attributes allowed by an object class structure. The `aci` attribute is multi-valued, which means that you can define several ACIs for the same entry or subtree.

Permissions

You allow or deny permissions. In general, you should avoid denying permissions for the reasons explained in "Allowing or Denying Access," on page 195.

You can allow or deny the following permissions:

- Read

  Indicates whether directory data may be read.

- Write

  Indicates whether directory data may be changed or created. This permission also allows directory data to be deleted, but not the entry itself. To delete an entire entry, the user must have delete permissions.

- Search

  Indicates whether the directory data can be searched. This differs from the read permission in that read allows directory data to be viewed if it is returned as part of a search operation. For example, if you allow searching for common names and read for a person's room number, then the room number can be returned as part of the common name search, but the room number cannot, itself, be searched for. This would prevent people from searching your directory to see who occupies a particular room.

- Compare

  Indicates whether the data may be used in comparison operations. Compare implies the ability to search, but actual directory information is not returned from the search. Instead, a simple Boolean value is returned that indicates whether the compared values match. This is used to match `userPassword` attribute values during directory authentication.

- Selfwrite

Used only for group management. This permission allows users to add or delete themselves from a group. Selfwrite works with proxy authorization: it grants the right to add or remove the proxy DN from a group entry (not the DN of the bound user).

- Add

  Indicates whether child entries can be created. This permission allows a user to create child entries beneath the targeted entry.

- Delete

  Indicates whether an entry can be deleted. This permission allows a user to delete the targeted entry.

- Proxy

  Indicates that the user can use any other DN (except Directory Manager) to access the directory with the rights of this DN.

Bind Rules

The bind rule usually indicates the bind DN subject to the permission. It can also specify bind attributes such as time of day or IP address.

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

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

- Only if the bind operation is arriving from a specific IP address or DNS hostname. This is often used to force all directory updates to occur from a given machine or network domain.
- If the person binds anonymously. Setting a permission for anonymous bind means that the permission also applies to anyone who binds to the directory.
- For anyone who successfully binds to the directory. This allows general access while preventing anonymous access.
- Only if the client has bound as the immediate parent of the entry.
- Only if the entry that the person has bound as meets a specific LDAP search criteria.

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

- Parent

If the bind DN is the immediate parent entry, then the bind rule is true. This allows you to grant specific permissions that, for example, allow a directory branch point to manage its immediate child entries.

- Self

  If the bind DN is the same as the entry requesting access, then the bind rule is true. For example, you can grant specific permission that allows individuals to update their own entries.

- All

  The bind rule is true for anyone who has successfully bound to the directory.

- Anyone

  The bind rule is true for everyone. This keyword is what allows or denies anonymous access.

Default ACIs

From an access control design perspective it is important to understand which default ACIs apply to the directory information you have stored in the `userRoot` database, so that you can then decide how best to tailor it to your deployment needs by modifying them. Whenever you create a new database in the directory, the top entry has the default ACIs listed below:

- Users can modify their own entry in the directory, but not delete it. They cannot modify the `aci` and `nsroledn` attributes.
- Users have anonymous access to the directory for search, compare, and read operations.
- The administrator (by default `uid=admin,ou=Administrators, ou=TopologyManagement,o=NetscapeRoot`) has all rights except proxy rights.
- All members of the Configuration Administrators group have all rights except proxy rights.
- All members of the Directory Administrators group have all rights except proxy rights.
- SIE group.

The `o=NetscapeRoot` subtree has its own set of default ACIs:

- All members of the Configuration Administrators group have all rights on the `o=NetscapeRoot` subtree except proxy rights.

- Users have anonymous access to the `o=NetscapeRoot` subtree for search and read operations.
- Entries under `o=NetscapeRoot` can define which groups have read, search, and compare access to them using the value of the `uniqueMember` attribute.
- All authenticated users have search, compare, and read rights to configuration attributes that identify the administration server.

For information on how to modify these default settings and implement your access control policy, see the *Sun ONE Directory Server Administration Guide*.

Deciding How to Set Permissions

If there are no ACIs present in the directory, then the default policy is *not* to grant users access rights of any kind. The exception to this is the directory manager. For this reason, you must set some ACIs for Directory Server if you want your users to be able to access your directory. The following sections describe the access control mechanism provided by Directory Server. For information about how to set ACIs, see the Managing Access Control chapter of the *Sun ONE 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. The rule states that when two conflicting permissions exist, the permission that denies access always takes precedence over the permission that grants access.

For example, if you deny write permission at the directory's root level, and you make that permission applicable to everyone accessing 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 in question.

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 privileges, you avoid the need to set an explicit deny privilege.

When to Deny Access

You rarely need to set an explicit deny. However, you may find an explicit deny useful in the following circumstances:

- You have a large directory tree with a complicated ACL spread across it.

  For security reasons, you find that you suddenly need to deny access to a particular user, group, or physical location. Rather than 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.

NOTE From a performance standpoint it is important to realize that ACIs are kept in memory and can considerably impact your memory usage performance. When a server starts all access controls are brought into memory, although cache limits do not apply to them. We recommend therefore, that in organizations using repeating directory tree structures, you optimize the number of ACIs used in Directory Server by using macro ACIs where possible.

Macros 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, or in the bind rule portion, or both. In practice, when Directory Server gets an incoming LDAP operation, the ACI macros are matched against the resource targeted by the LDAP operation. If there is a match, the macro is replaced by the value of the DN of the targeted resource. Directory Server then evaluates the ACI normally. For more information on macro ACIs refer to the Managing Access Control chapter of the *Sun ONE Directory Server Administration Guide.*

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 allow you to set access to any directory entry that matches a defined set of criteria.

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

Filtered access control rules let you use predefine levels of access. For example, suppose your directory contains home address and telephone number information. Some people want to publish this information, while others want to be “unlisted.” You can handle this situation by doing the following:

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

For more information about using LDAP search filters, and on using LDAP search filters with ACIs, see the *Sun ONE Directory Server Administration Guide.*

Requesting Effective Rights Information

The rich access control model provided by Directory Server is powerful in that access can be granted to users via many different mechanisms, but this richness, although very welcome when you are having to configure your access control policy, means that it can be a complex affair trying to determine what the policy actually comprises at a later date. Because there are several parameters that can define the security context of a user, for example, IP address and machine name, time of day, authentication method, or simply the type of attribute you may be trying to add, the need to be able to list the entry and attribute permissions becomes critical. Without this ability to request the rights of a given user to directory entries and attributes, user administration, access control policy verification, and debugging would be much more difficult.

Sun ONE Directory Server 5.2 provides a new feature called Effective Rights, which allows clients to query what access control rights they have to directory entries and attributes.

NOTE It is important to understand that the effective rights information you obtain with the Effective Rights control 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 trying to establish why a given user does or does not have access to certain data, system administrators or others requesting the effective rights will have to be sure to reflect all of the user's parameters when they initiate their effective rights search operation.

This section examines the effective rights feature in more detail and is divided into the following sub-sections:

- About the Effective Rights Feature
- Access Control on the Effective Rights Feature
- Understanding the Effective Rights Results

About the Effective Rights Feature

The effective rights feature works via an `ldapsearch` operation. Note that this feature requires the `ldapsearch` utility supplied with the Directory Server Resource Kit (DSRK).

The rights information you require is specified using a particular option (discussed below) and the information relative to these rights is returned with your `ldapsearch` results. To specify the rights information you require, use the following elements in your `ldapsearch` operation:

- An Effective Rights control (`1.3.6.1.4.4.42.2.27.9.5.2`) to specify that you are requesting effective rights information. You include this control in your `ldapsearch` operation using the `-J` option.
- Arequest for an operational attribute `aclRights` to return access control effective rights information at both an entry and attribute level.
- The `-c` option of `ldapsearch` to specify the user whose rights you are requesting

- If necessary, the `-X` option of `ldapsearch` to specify the list of attribute types for which you want to have rights information, if they are not already being returned with the entry.

NOTE It is only necessary to specify the Effective Rights control if you are not using the `-c` or `-X` options in your `ldapsearch` operation, as either of these options will automatically attach the control to your search.

Also, if there is a NULL value for the Effective Rights control then Directory Server interprets that to mean that you want to retrieve both the rights for the current user and rights for the attributes and entries being returned with the current `ldapsearch` operation.

The first question you need to ask yourselves when using the effective rights feature is for which user you are trying to obtain the effective access control rights. Once you know which user's rights you require, you can then specify that user with the `-c ldapsearch` option. It is worth noting that if you include

```
-c "dn:"
```

in your `ldapsearch` command, you are requesting effective rights for an anonymous user, and if you have NULL or an empty string for the `-c` option, that is `-c ""`, then you are requesting the bound user's rights.

The next question you need to ask yourself is for which attributes you are requesting the effective rights. The attributes for which you require rights information do not necessarily need to exist in the entry. We can imagine the scenario where an entry does not include the `description` attribute, but that you want to know which rights a given user would have on that attribute. To specify the list of attribute types for which you want rights information, you use the `-X ldapsearch` option followed by the attribute name, for example `-X description`.

NOTE As far as the other parameters that affect access control are concerned, such as time of day, authentication method, machine address and name, the functionality simply inherits these parameters from the user initiating the search operation.

The effective rights feature provided with Directory Server also allows you to specify different levels of access control information. You can choose either to request only the rights information, only the logging information, or both together. Since the logging information (which includes a synopsis of the rights information)

provides detail down to a permission level, it allows you not only to establish what access control policy is effective, but also to understand why a given operation may have been denied or allowed. The subtyping for the operational `aclRights` attribute that allows you to specify the information you want to have returned is as follows:

`aclRights`	Requests the rights information only
`aclRightsInfo`	Requests the logging information only
`aclRights aclRightsInfo`	Requests both the rights and logging information

CAUTION You cannot use the `aclRights` attribute in a search filter.

Access Control on the Effective Rights Feature

In order to successfully obtain the effective rights information, users must have the access control rights to use the Effective Rights control, *and*, in order to have any rights information returned to the user for a given entry, the user will also need read access to the `aclRights` attribute. This double layer of access control for the effective rights feature provides basic protection which can then be more finely tuned where necessary. By analogy with proxy, if you have read access to the `aclRights` attribute in an entry, then you can ask about anyone's rights to that entry and its attributes. The logic behind this access control setup is that it makes sense for the user who manages the resource to know who has rights to that resource, even if that same 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, then the operation will fail and an error message to that effect will be sent. However, if the user requesting rights information does have the rights to use the control but lacks the rights to read the `aclRights` attribute, then the `aclRights` attribute will simply be absent from the returned entry. This behavior reflects the Directory Server's general search operation behavior.

NOTE If a proxy control is attached to a Effective Rights control-based `ldapsearch` operation, then the effective rights operation will be authorized as the proxy user. This means that it is the *proxy user* which will require rights to use the Effective Rights control, and that the entries returned will be those entries that the *proxy user* has the right to search and see.

Understanding the Effective Rights Results

This part presents the effective rights search results and is divided into the following subparts:

- Rights Information
- Write, Selfwrite_add, and Selfwrite_delete permissions
- Logging Information

Rights Information

The effective rights information is presented according to the following subtypes:

`aclRights;entrylevel`	Presents the entry level rights information
`aclRights;attributelevel`	Presents the attribute level rights information
`aclRightsInfo;entrylevel`	Presents the entry level logging information
`aclRightsInfo;attributelevel`	Presents the attribute level logging information

Once the information has been presented down to this level then the permissions and attribute names are used to subtype and separate the information.

For Sun ONE Directory Server 5.2 the `aclRights` string appears as follows:

*permission:value(permission:value)**

Possible entry level permissions are `add`, `delete`, `read`, `write`, and `proxy` while possible attribute level permissions are `read`, `search`, `compare`, `write`, `selfwrite_add`, `selfwrite_delete`, and `proxy`.

CAUTION In future releases of Directory Server new permissions may be added to this string.

The `aclRights` string containing the results of the requested rights information will either mark these permissions as:

- "`0`" for not granted
- "`1`" for granted or
- "`?`" for cases where the granting of rights depends on the value of the attribute you are adding, deleting, or replacing. If you see a `?`, then you are advised to consult the logging information to establish exactly why the permissions will or will not be granted.
- "`-`" to indicate that the attribute in question is a virtual attribute, and therefore not updatable. The only way to modify a virtual attribute is to modify the mechanism that generates it.

By way of an example, imagine that you wanted to request the effective rights (excluding the logging information) for the `cn` attribute of the user `cn=justread` in the `cn=peopletestResource` subtree. You would run the following `ldapsearch` command:

```
./ldapsearch -D "cn=directory manager" -w secret12
-c "dn:cn=justread,dc=france,dc=sun,dc=com" -p 5200
-b cn=peopletestResource,dc=france,dc=sun,dc=com
objectclass=* cn "aclRights"
```

If the user `cn=justread` has only read permissions on the `cn` attribute, then the requested effective rights information would appear as follows:

```
dn: cn=peopleTestResource,dc=france,dc=sun,dc=com
aclRights;attributeLevel;cn:search:0,read:1,compare:0,write:0,
selfwrite_add:0,selfwrite_delete:0,proxy:0
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
```

Write, Selfwrite_add, and Selfwrite_delete permissions

In Sun ONE Directory Server 5.2 only the write attribute level permissions can be marked with a "`?`", although this may change in future releases. For the add and delete permissions, the entries you can add and delete can depend on the values of the attributes in the entry. However, instead of returning a "`?`" should it prove to be the case, the permission (i.e. "`0`" or "`1`") is returned on the entries as they are returned with the `ldapsearch` operation.

If the value for a `write` permission is "`1`", then this means that the permission is granted for both `add` and `delete` `ldapmodify` operations for all values except possibly the authorization `dn` value. A value of `0` for a write permission means that the permission is not granted for either `add` or `delete` `ldapmodify` operations for any values, except possibly the value of the authorization `dn`. The permission in force for the value of the authorization `dn` is returned explicitly in one of the `selfwrite` permissions; that is, either `selfwrite_add` or `selfwrite_delete`.

It is necessary to make a distinction between `selfwrite-add` and `selfwrite-delete` attribute level permissions in the effective rights functionality. Although these permissions do not actually exist as such in the context of ACIs, the reality of a set of ACIs can be to grant a user `selfwrite` permission for *just* the `add` or *just* the `delete` part of a `modify` operation. The same distinction is not made for the `write` permission because, in contrast to the `selfwrite` permission where the value of the attribute being modified is defined, in that it is the authorization `dn`, the value of the attribute being modified for a write permission is undefined. When the effective permission depends on a `targattrfilters` ACI, we cannot extend the analysis further, and therefore use the "`?`" value to indicate that the logging information should be consulted for more permission detail. Given the relative complexity of the interdependencies between the write, selfwrite_add, and selfwrite_delete permissions, Table 7-5 on page 204 explains what the possible combinations of these three permissions actually mean.

Table 7-5 Effective Rights Interdependencies Between write, selfwrite_add, and selfwrite_delete Permissions

write	selfwrite_add	selfwrite_delete	Effective Rights Explanation
0	0	0	Cannot add or delete any values of this attribute.
0	0	1	Can only delete the value of the authorization dn.
0	1	0	Can only add the value of the authorization dn.
0	1	1	Can only add or delete the value of the authorization dn.
1	0	0	Can add or delete all values except the authorization dn.
1	0	1	Can delete all values *including* the authorization dn and can add all values *excluding* the authorization dn.
1	1	0	Can add all values *including* the authorization dn and can delete all values *excluding* the authorization dn.
1	1	1	Can add or delete all values of this attribute.
?	0	0	Cannot add or delete the authorization dn value, but may be able to add or delete other values. Refer to logging information for further detail regarding the write permission.

Table 7-5 Effective Rights Interdependencies Between write, selfwrite_add, and selfwrite_delete Permissions

write	selfwrite_add	selfwrite_delete	Effective Rights Explanation
?	0	1	Can delete but cannot add the value of the authorization dn, and may be able to add or delete other values. Refer to logging information for further detail regarding the write permission.
?	1	0	Can add but cannot delete the value of the authorization dn, and may be able to add or delete other values. Refer to logging information for further detail regarding the write permission.
?	1	1	Can add and delete the value of the authorization dn, and may be able to modify add or modify delete other values. Refer to logging information for further detail regarding the write permission.

Logging Information

The effective rights logging information provides you with the key to understanding, and therefore being able to debug access control difficulties. The logging information contains an access control summary statement, called the `acl_summary`, that provides you with the reasons as to why your access control has either been allowed or denied. The access control summary statement tells you the following:

- Whether access was allowed or denied
- The permissions granted
- The target entry of the permissions
- The name of the target attribute
- The subject of the rights being requested
- Whether or not the request was made by proxy, and if so, the proxy authentication DN
- And, most importantly, for debugging purposes, the reason for allowing or denying access. The possible reasons are listed in Table 7-6 on page 205:

Table 7-6 Effective Rights Logging Information Reasons and Their Explanations

Logging Information Reason	Explanation of reason
`no reason available`	No reason available to explain why access was allowed or denied.

Table 7-6 Effective Rights Logging Information Reasons and Their Explanations

Logging Information Reason	Explanation of reason
`no allow acis`	No allow ACIs exist which results in denied access.
`result cached deny`	Cached information was used to determine the access denied decision.
`result cached allow`	Cached information was used to determine the access allowed decision.
`evaluated allow`	An ACI was evaluated to determine the access allowed decision. The name of the deciding aci is included in the log information.
`evaluated deny`	An ACI was evaluated to determine the access denied decision. The name of the deciding aci is included in the log information.
`no acis matched the resource`	No ACIs match the resource or target, which results in denied access.
`no acis matched the subject`	No ACIs match the subject requesting access control, which results in denied access.
`allow anyone aci matched anon user`	An ACI with a userdn = `"ldap:///anyone"` subject allowed access to the anonymous user.
`no matching anyone aci for anon user`	No ACI with a userdn= `"ldap:///anyone"` subject was found, and so access for the anonymous user was denied.
user root	The user is root DN and is allowed access.

NOTE Write permissions for virtual attributes are not provided, nor is any associated logging evaluation information, because virtual attributes are not updatable.

For the actual log file format see *Sun ONE Directory Server Reference Manual.*

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. They are included here to provide you with 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, because ACIs are kept in memory, overly large numbers of ACIs can make inefficient use of your ACI memory. A large number of ACIs makes it hard for you to determine immediately the directory object available to particular clients. Reducing the number of ACIs in your directory tree, and preferring the use of macro ACIs where possible, will not only make it easier to manage your access control policy, but will also improve the efficiency of ACI memory usage.

- Balance allow and deny permissions.

  Although the default rule is to deny access to any user who has not been specifically granted access, you might find that you can save on the number of ACIs by using one ACI allowing access close to the root of the tree, and a small number of deny ACIs close to the leaf entries. This scenario can avoid the use of multiple allow ACIs close to the leaf entries.

- Identify the smallest set of attributes on any given ACI.

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

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

- Use LDAP search filters cautiously.

  Because search filters do not directly name the object that you are managing access for, their use can result in unexpected surprises, especially as your directory becomes more complex. If you are using search filters in ACIs, run an `ldapsearch` operation using the same filter to make sure you know what the results of the changes mean to your directory.

- Do not duplicate ACIs in differing parts of your directory tree.

Watch out for overlapping ACIs. For example, if you have an ACI at your directory root point that allows a group write access to the `commonName` and `givenName` attributes and another ACI that allows the same group write access for just the `commonName` attribute, then consider reworking your ACIs so that only one control grants the write access for the group.

As your directory grows more complicated, it becomes increasingly easy to accidentally overlap ACIs in this manner. By avoiding ACI overlap, you make your security management easier while potentially reducing the total number of ACIs contained in your directory.

- Name your ACIs.

  While naming ACIs is optional, giving each ACI a short, meaningful name helps you to manage your security model, especially when examining your ACIs from the Directory 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 70.

- Group your ACIs as closely together as possible within your directory.

  Try to limit ACI placement to your directory root point and to major directory branch points. Grouping ACIs helps you manage your total list of ACIs, 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 is confusing for a human administrator.

ACI Limitations

When creating an access control policy for your directory service, you need to 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, then 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 Database Links and Access Control Evalutation in the *Sun ONE 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. For more information on this keyword, refer to Chapter 4, "Designing the Directory Tree."
- 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 at all. For more information on LDAP URLs, see LDAP URLs in the *Sun ONE Directory Server Reference Manual*.
- 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

After designing your authentication scheme for identified users and your 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.

By having separate ports, 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, called a *cipher*, in 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

For more information about ciphers and hashing algorithms, refer to "Other Security Resources," on page 220.

After encryption of the 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.

Now the connection is secure and the server is authenticated to the client. You may configure the server to further request authentication from the client. Directory Server supports certificate-based and SASL-based client authentication. These mechanisms are further described in "Selecting Appropriate Authentication Methods," on page 169. 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 B, "Using a Sun Crypto Accelerator Board" in the *Sun ONE Directory Server Installation and Tuning Guide*.

For information about configuring and enabling SSL in both Directory Server and its clients, refer to Chapter 11, "Implementing Security" in the *Sun ONE Directory Server Administration Guide*.

Encrypting Attributes

This section presents the attribute encryption functionality new to Sun ONE Directory Server 5.2 and is divided into the following sub-sections:

- 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 new attribute encryption feature prevents users from accessing this sensitive data while it is in storage.

Attribute encryption allows 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 particular 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.

NOTES

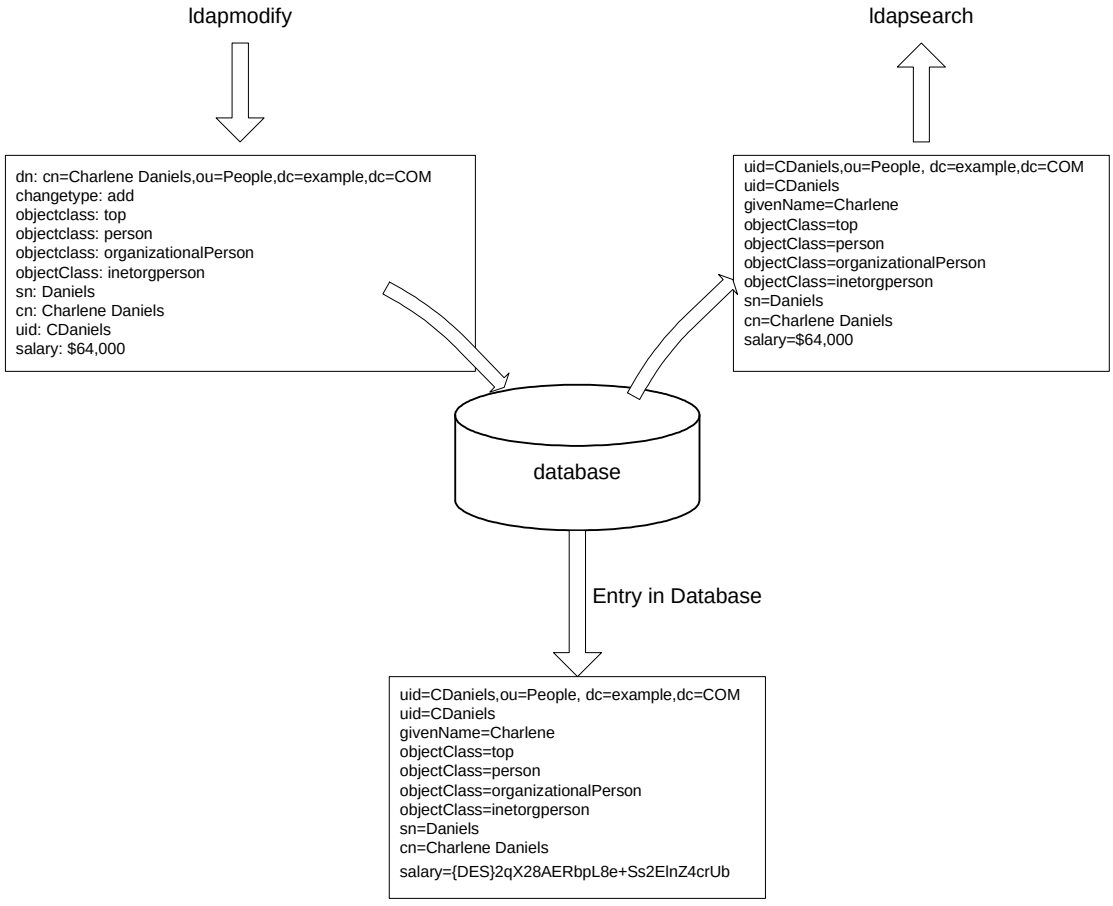
1. Although attribute encryption supports the encryption of the `userPassword` attribute, we strongly recommend that you do NOT use this functionality as the sole means of protecting user passwords. Ensure that access control to the `userPassword` attribute is sufficiently protected. In addition, you should assign an appropriate password policy to the `userPassword` attribute, to benefit from the security provided by the imposed password length, validity, expiration warnings, checks, history and reset options.
2. If you specify an RDN attribute for encryption, the DN will not be encrypted. Attribute encryption does not apply to DNs.

When in storage, encrypted attributes are prefaced with a cipher tag that indicates the encryption algorithm used. An encrypted attribute using the DES encryption algorithm would appear as follows:

```
{CKM_DES_CBC}3hakc&jla+=snda%
```

In addition to protecting data while in storage, attribute encryption also allows 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 on page 213 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



Attribute Encryption Implementation

The Directory Server attribute encryption feature supports a wide range of encryption algorithms, and ensures portability across different platforms. The required encryption algorithm is specified using the `dsEncryptionAlgorithm` attribute. See the *Sun ONE Directory Server Reference Manual* for further information regarding attribute encryption configuration attributes.

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 your data (a more security sensitive operation), Directory Server automatically prompts you for the key database password, irrespective of whether your export operation be online or offline. This provides an additional security layer.

NOTE As long as the certificate 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

The following list of usage considerations outlines some of the items you must take into account when using the attribute encryption feature:

- As a general best practice when making attribute encryption configuration changes, we recommend that you export your data, make the configuration changes, and then import the newly configured data.

  Adopting such a practice will ensure that all configuration changes are taken into account in their entirety, without any loss in functionality. Failing to adopt such a practice 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 at a later stage that you want 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 `mySensitiveAttribute` 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.

- Users can either enter passwords directly or enter a path to the file containing the password that has the same syntax as the SSL password file.

  For more information concerning attribute encryption procedures, see “Encrypting Attribute Values” in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.

- Algorithm changes are supported, but can result in lost indexing functionality, if they are not made correctly.

  To change the algorithm used to encrypt your data, export your data, make the attribute encryption configuration changes, and then import your 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. Therefore, Directory Server provides additional security as it 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 your 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 be able to decrypt encrypted data. Changing the certificate without decrypting the data beforehand will result in you no longer being able to decrypt your data. Once again we recommend the best practice of exporting your data in decrypted format, changing your certificate and then re-importing your data.

- To transport data in encrypted format, that is, export and import it from one server instance to another, both server instances must have used the same certificate.

Grouping Entries Securely

This section deals with the security issues related to grouping entries securely and contains the following sections:

- 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 easily 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 the section on Inactivating Users and Roles in the *Sun ONE 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 be sure to protect the part of the filter that would prevent the user from being able to relinquish the filtered role by modifying an attribute. The user 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 "Managing Access Control" in the *Sun ONE Directory Server Administration Guide*.

Protecting the CoS Definition Entry

Even though 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 will need to 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 regarding these procedures see the *Sun ONE 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

Finally, 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 216.

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 (which is the entry that is 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, take a look at the following:

- Sun ONE Middleware Developer Security Resources
  http://developer.iplanet.com/tech/security/
- *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

Monitoring Your Directory

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 Sun ONE Directory Server.

NOTE This chapter does *not* focus on system and network monitoring, as this is an area not specific to Sun ONE Directory Server.

This chapter is divided into the following sections:

- Defining a Monitoring and Event Management Strategy
- Directory Server Monitoring Tools
- Directory Server Monitoring
- SNMP Monitoring

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 for defining an effective monitoring can be broken down into the following steps:

1. Select the appropriate monitoring tools, whether they be operating system tools, Sun ONE Directory Server monitoring tools, or third party monitoring tools.
2. Identify the key performance measures 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 using the monitoring tools to monitor 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 Sun ONE 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. For information on the Sun ONE scripts that can be used to access logging information, refer to the *Sun ONE DSRK Tools Reference*. For information on viewing and configuring log files refer to the Managing Log Files chapter in the *Sun ONE Directory Server Administration Guide.*

- Directory Server Console

  The Sun ONE Directory Server console is useful for monitoring 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 Sun ONE 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 231. For information on how to set up and configure SNMP refer to the Monitoring Directory Server Using SNMP chapter of the *Sun ONE Directory Server Administration Guide*.

Directory Server Monitoring

The most important step in defining a monitoring and event management strategy is determining the key performance measures 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

NOTES

- 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 in order to access the information. Having such permissions is therefore a prerequisite in defining your monitoring strategy
- Although it is essential to monitor the operating system environment upon which Sun ONE Directory Server is running to ensure that the system is running efficiently and not compromising Directory Server, this area is *not* covered in this chapter as it is not specific to Sun ONE 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 the Sun ONE 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.

- The 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 From the Directory Server console" in the *Sun ONE Directory Server Administration Guide.*
- Running an `ldapsearch` command on the `cn=monitor,cn=config` 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 more detail regarding the performance counters stored under `cn=monitor,cn=config`, refer to "Monitoring Your Server From the Command Line" in the *Sun ONE Directory Server Administration Guide.*
- On UNIX systems, 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 further information or the equivalent command line tool documentation provided with your operating system.
- 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 Directory Server is running. For more information on access log content and connection codes refer to the Access Logs and Connection Codes section of the *Sun ONE Directory Server Reference Manual.*

- 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 "Managing Log Files" in the *Sun ONE 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 the Core Server Configuration and the Plug-in Implemented Server Functionality chapters of the *Sun ONE Directory Server Reference Manual.*

  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 “Core Server Configuration Attributes” and “Plug-in Implemented Server Functionality”in the *Sun ONE Directory Server Reference Manual*.

- 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.

For more information on the `cn=disk,cn=monitor` attributes as well as the configurable disk low or full thresholds, refer to the “Core Server Configuration Attributes” and “Plug-in Implemented Server Functionality” in the *Sun ONE Directory Server Reference Manual*.

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.

Sun ONE 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 between a master replica and one or more consumer replicas. An awareness of the level of synchronization is vital when it comes to managing potential conflicts.

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 the Monitoring Replication Status section of the *Sun ONE Directory Server Administration Guide.*

For more information about the replication monitoring tools, refer to the Replication Monitoring Tools section of the *Sun ONE Directory Server Reference Manual.*

Monitoring Indexing Efficiency

Indexing impacts write performance (when creating indexes) and read performance (when searching the directory.) It is therefore important to monitor indexing efficiency to maintain an appropriate balance between write and read 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. 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 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 the Access Logs and Connection Codes section of the *Sun ONE Directory Server Reference Manual*. For an complete list of Directory Server configuration attributes, refer to the Core Server Configuration and the Plug-in Implemented Server Functionality chapters of the *Sun ONE Directory Server Reference Manual*.

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 new 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 198.

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 Sun ONE 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. Sun ONE 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, Enterprise Server, and Messaging Server are all installed on the same host, the subagents for each of these servers communicates with the same master agent. In the Windows environment, the master agent is the SNMP service provided by the Windows operating system. In the UNIX environment, the master agent is installed with Sun ONE 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

Sun ONE 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 Sun ONE Directory Server

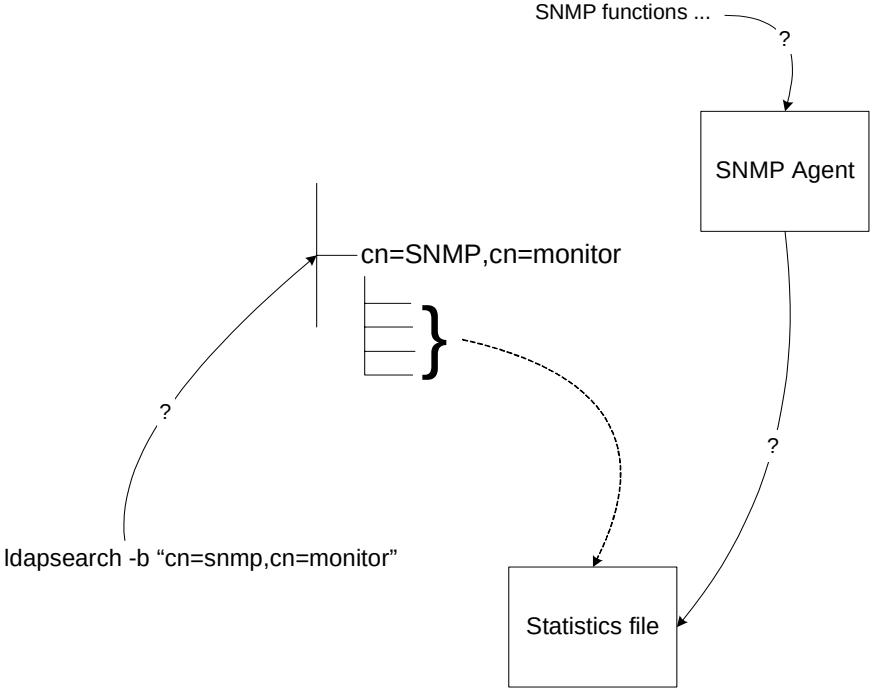
Directory Server supports SNMP monitoring in two ways:

- 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 the Monitoring Directory Server Using SNMP chapter of the *Sun ONE Directory Server Administration Guide.*

The SNMP managed objects supported by Sun ONE 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 the "Monitoring Attributes" section of the *Sun ONE Directory Server Reference Manual.* Names of attributes returned by the SNMP agent are prefixed with `ds`.

In addition to the operations managed objects, Sun ONE Directory Server supports managed objects related to the interactions between the monitored Directory Server and its peer Directory Servers. These objects are listed in Table 8-2:

Table 8-2 Interactions Table of Supported SNMP Managed Objects

Managed Object	Description
dsTimeOfCreation	The value of system "up" time when the entry containing interaction details of (attempted) interaction between the Directory Server and a peer Directory Server was created. If the entry was created before the management network subsystem was initialized, this object will contain a value of zero.
dsTimeOfLastAttempt	The value of system "up" time when the last attempt was made to contact this Directory Server. If the last attempt was made before the network management subsystem was initialized, this object will contain a value of zero.
dsTimeOfLastSuccess	The value of system "up" time when the last attempt made to contact this Directory Server was successful. If none of the attempts have been successful, this object will have a value of zero. If the last successful attempt was made before the network management subsystem was initialized, this object will contain a value of zero.
dsFailuresSinceLastSuccess	The number of failures since the last successful attempt to contact this Directory Server. If there have been no successful attempts, this object will contain the number of failures since this entry was created.
dsFailures	Cumulative failures to contact the peer Directory Server since the creation of this entry.
dsSuccesses	Cumulative successes since the creation of this entry.
dsURL	URL of the peer Directory Server.

Sun ONE Directory Server also supports entity related managed objects, containing information about the current installation of Directory Server. These managed objects are listed in Table 8-3.

Table 8-3 Entity Table of SNMP Supported Managed Objects

Managed Object	Description
dsEntityDescr	A general textual description of the installed Directory Server.
dsEntityVers	Directory Server version.
dsEntityOrg	Organization responsible for this installation of Directory Server.
dsEntityLocation	Physical location of this Directory Server. For example: hostname, building, number, laboratory number, etc.
dsEntityContact	Contact person responsible for the installed Directory Server and their contact details.
dsEntityName	Name assigned to the installation of Directory Server by the installation site.

Part 2

Directory Server Deployment Scenario and Reference Architectures

The second part of this guide presents a Sun ONE Directory Server 5.2 deployment scenario and outlines a number of architectural strategies. Presented from a business perspective, the deployment scenario may help you understand how Sun ONE Directory Server 5.2 can be deployed to provide solutions to a given business challenge. Each architectural strategy, categorized by the number of data centers (sites) in which your organization has a directory, provides guidelines regarding the physical location of your data, an appropriate replication topology, in addition to suggestions related to failover, scalability and backup strategies.

While we advise that you implement your Directory Server deployment with the help of Sun Professional Services, the following deployment scenario and architectural strategies should give you an insight into some of the major concerns and issues you will need to address.

Chapter 9

Banking Deployment Scenario

Business Challenge

ExampleBank is an international bank which wishes to provide its customers and employees with e-banking and phone banking facilities. Customers may, for example, want to check their bank account balance, perform intrabank and interbank money transfers, check investment portfolios, set up meetings with bank managers, access the bank's online news service and change profile information, while bank employees may want to login to work from home, perform lookups on the bank's phone book, change profile information or perform bank transactions for customers using the bank's phone banking service. The business challenge for ExampleBank is to allow its users, customers and employees alike, access to their range of banking services in a way that guarantees:

- uncompromised security,
- good performance,
- scalability,
- and manageability.

To rise to this business challenge, ExampleBank has deployed Sun ONE Directory Server 5.2 as the cornerstone of its banking deployment. Directory Server holds all of ExampleBank's user authentication and profile data which is used each time a user authenticates to the bank's online system whether it be via a portal or over the phone. The following deployment scenario details how ExampleBank has implemented Directory Server to achieve their e-banking and phone banking business objectives and is divided into the following sections:

- Deployment Context and Replication Topology
- Performance Requirements
- Schema, Data, and Directory Information Tree Design

- Security Considerations
- Implementation

Deployment Context and Replication Topology

Deployment Context

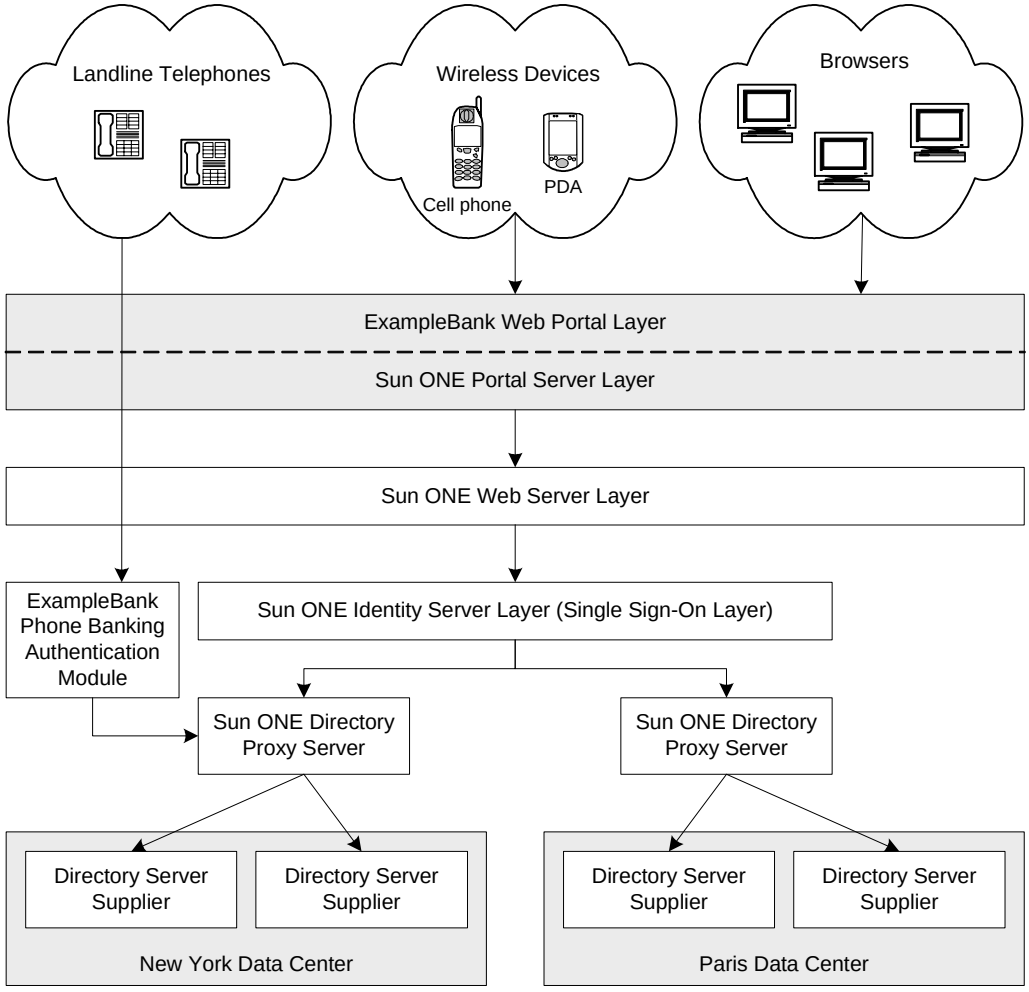
ExampleBank's deployment needs to be able to cater for both the e-banking and phone banking services and ensure that authentication to both services is secure, manageable, swift and scalable. ExampleBank uses the same stack of products to cater for both types of access which we examine a little closer in the following paragraphs.

To access the e-banking facilities ExampleBank users use amongst other things, Sun ONE Portal Server. The portal runs over Sun ONE Web Servers coupled with the Sun ONE Identity Server which offers ExampleBank the single sign-on solution it needs for login manageability. ExampleBank has placed Sun ONE Directory Proxy Server next in line in the deployment stack, as the Directory Proxy Server, with its ability to manage referrals to chosen servers caters for Directory Server's operational flow load distribution and transparent failover needs. At the bottom of the deployment stack sits Directory Server, which holds the user profile and authentication data necessary to conduct banking business.

As regards access to the phone banking facilities ExampleBank users phone a given number and enter their account number followed by their phone banking pin number. A special phone banking authentication module contacts Directory Server to try to find the account number, then verifies if the pin number is correct, and if it is correct, forwards the call to a member of the phone banking team.

ExampleBank's banking deployment stack is illustrated in Figure 9-1 on page 241:

Figure 9-1 ExampleBank's Banking Deployment Stack



Replication Topology

This section outlines ExampleBank's replication strategy and is divided into the following parts:

- Replication Topology Overview
- Failure and Recovery Scenarios

Replication Topology Overview

ExampleBank has two major data centers, one in New York and the other in Paris. Because it is essential that services remain accessible 24 hours a day and that local write failover needs be catered for, the replication design choice is to have a pair of masters in each data center. This 4-way master replication topology across two continents in ExampleBank's deployment is made possible by the new 4-way Multi-Master Replication (MMR) over WAN possibilities offered by Sun ONE Directory Server 5.2. Hubs have also been included in the replication topology to facilitate load distribution, and the four consumers in each data center allow this topology to scale for read (lookup) operations. ExampleBank's replication topology for its two, cross-continent data center deployment, is illustrated in Figure 9-2 on page 242:

Figure 9-2 Example Bank's Two Data Center Replication Topology

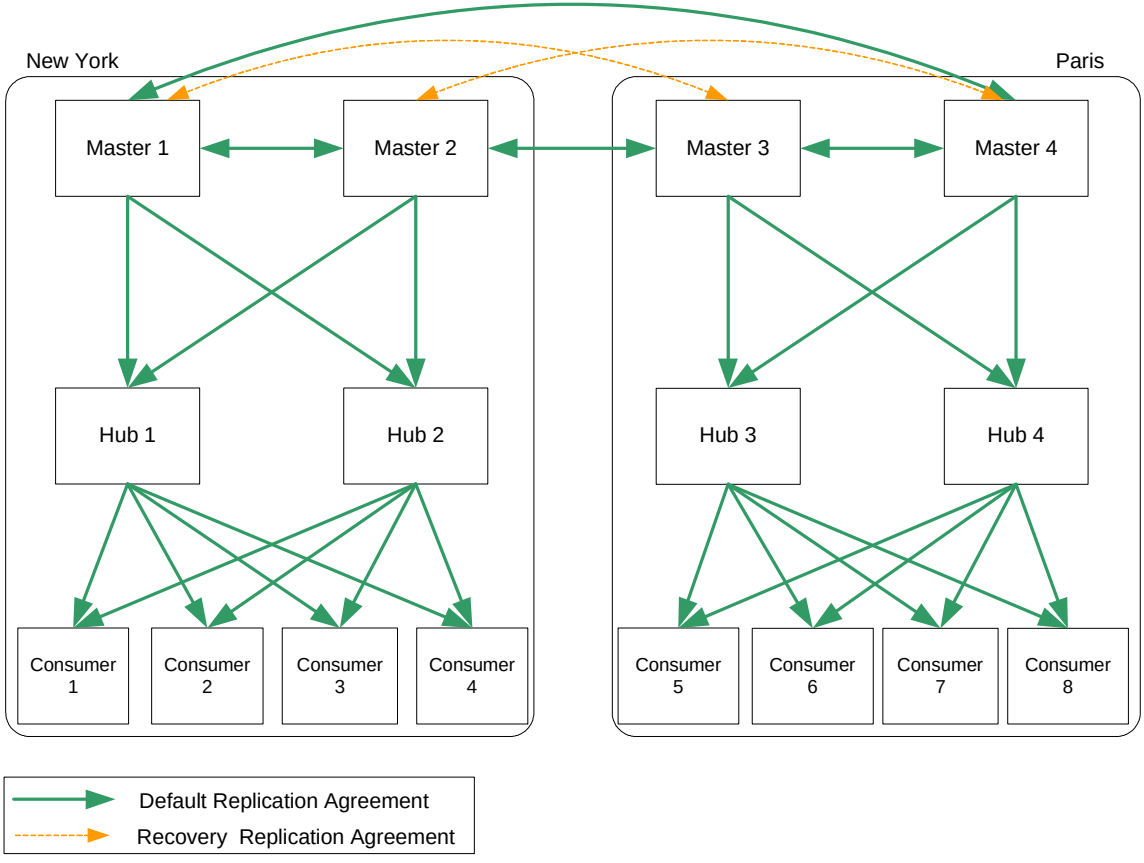


Figure 9-2 on page 242 shows the replication agreements which are enabled by default, between masters and hubs, and hubs and consumers, in addition to two recovery replication agreements between masters 1 and 3 and masters 2 and 4, that could be enabled should a master go offline. Sun ONE Directory Server 5.2 allows you to configure replication agreements and then enable or disable them as and when you need to, which offers a welcome degree of topology flexibility. This not only allows you to prepare for recovery scenarios, but, should you wish to do so, also allows you to optimize network usage and minimize the number of modifications actually being sent at any given time.

NOTE We recommend the use of multiple WAN connections for the inter-data center links to maintain optimal availability.

Failure and Recovery Scenarios

Any number of failures can occur which compromise your deployment, be they related to network links, the Directory Server process (`slapd`), hardware, replication failure or an overburden on the system due to an excess of read or write operations. It is vital that your replication topology is configured in such a way as to provide the necessary recovery solutions.

In the event of one master going offline, you would simply enable the appropriate recovery replication agreement (which you would already have configured but left disabled) as indicated in Figure 9-2 on page 242. However, should two masters go offline in the same data center, then you have a number of options open to you. One recovery solution might be to enable a previously configured replication agreement between one of the masters in the unaffected data center and one of the hubs in the impacted data center. Another recovery solution might be to take advantage of Sun ONE Directory Server 5.2's online replica promotion and demotion feature, to promote a hub in the impacted data center to master to provide local write coverage until the impacted masters are brought back online. For further information related to sample replication topologies, failure recovery, and backup strategies see Chapter 10, "Architectural Strategies". For further procedural information related to replication see the Managing Replication chapter in the *Sun ONE Directory Server Administration Guide*.

Performance Requirements

Performance is extremely important for ExampleBank as the viability of its e-banking and telephone banking offer depends largely on the rapidity of its operations. In addition to good performance, ExampleBank needs the foundation of its online banking stack to be scalable, as its expected annual growth rate stands at 6%. Directory Server caters for both of these requirements.

This section outlines the performance requirements of ExampleBank and is divided into the following subsections:

- User Demands
- Hardware Guidelines

User Demands

ExampleBank's current user base is 10 million users, of which 10, 000 are employees. The employee and customer user requirements are described in Table 9-1. It is on the basis of these requirements that ExampleBank will calculate the number of writes and reads per second that its online banking stack will have to manage.

Table 9-1 ExampleBank's User Demands

User Demand	Percentage of Users Involved	Frequency of Demand	Demands per second
Customer Login	70%	4 logins per week	195 logins
Employee Login	100%	3 logins per week	2 logins
Customer Banking Transaction	100%	1 transaction per week	70 transactions
Customer Account Balance Consultation	70%	3 balance consultations per month	25 transactions
User Profile Change	100%	1 data change per month	20 profile changes
Employee Lookup	100%	1 lookup per day	1 lookup
Employee performing customer transaction	50%	10 transactions per day	8 transactions

The transactions described in the preceeding table translate into LDAP operations as indicated in Table 9-2:

Table 9-2 User Demands and their LDAP Operation Equivalent

User Demand	LDAP Operation Equivalent
Login	Search + Bind + Search
Banking Transaction	Search + Modify
Balance Consultation	NOT AN LDAP OPERATION
Profile Change	Search + Modify
Lookup	Search

With this in mind, we can conclude that ExampleBank will need to be able to sustain the following in terms of search, bind and modify operations:

- 687 search operations
- 197 bind operations
- 98 modify operations

Hardware Guidelines

To give you a basic idea of hardware, for this type of deployment, which could be termed as a medium to large-scale deployment, we suggest the following machine type comprising:

- 8 CPUs,
- 32 to 64 GB RAM,
- 3 disk arrays configured as RAID, 655 GB, with for example:
  - database on first array
  - transaction logs on second array
  - changelog, access, and error logs on third array

Schema, Data, and Directory Information Tree Design

This section examines ExampleBank's schema, data, and directory information tree design in more detail and is divided into the following subsections:

- Schema
- Data
- Directory Information Tree

NOTE The schema and Directory Information Tree provided in this section are examples of what you may want to implement and do not constitute an exhaustive or definitive list. Some of the object classes and attributes are required by Sun ONE Identity Server and Sun ONE Portal Server. Refer to the Sun ONE Identity Server and Sun ONE Portal Server documentation for more information.

Schema

ExampleBank needs schema which describes its user profiles, e-banking services, and phone banking services. This section describes the schema, and is divided into two subsections:

- Attributes
- Object Classes

Attributes

ExampleBank only distinguishes between customers and employees at an attribute level using the `ebStatus` attribute.The attributes relative to user profiles, banking services and additional portfolio management banking services are listed below in the following two tables.

Table 9-3 lists the basic user profile attributes:

Table 9-3 ExampleBank's Basic User Profile Attributes

Attribute Name	Attribute Description	Syntax	Multivalued Attribute
ebID	ExampleBank unique identifier for users (both customers *and* employees)	Integer	No
ebStatus	Specifies whether or not a user is a customer, employee or contractor.	Directory String	Yes
ebPreferredLanguage	ExampleBank user's preferred language	Directory String	No
ebSecondaryLanguage	ExampleBank user's secondary language	Directory String	No
ebAreaCode	ExampleBank user's area code	Directory String	No
ebCurrentAccountNumber	ExampleBank user's current account number	Integer	No
ebSavingsAccountNumber	ExampleBank user's savings account number	Integer	No
ebPhoneBankingPin	Specifies the phone banking pin number	Integer	No
ebNewsLetterSubscription	Specifies if user subscribes to Example Bank's newsletter	Directory String	No
ebNewsLetterType	Specifies which newsletter the user subscribes to	Directory String	No
ebEBankingPreferencesFont	Specifies the user's preferred e-banking font	Directory String	No
ebEbankingPreferencesFontSize	Specifies the user's preferred e-banking font size	Directory String	No
ebPhoneBankingSafeNumber	Specifies a safe, well-known landline telephone number	Telephone Number	No

In addition to ExampleBank's user profile attributes, the attributes presented in Table 9-4 specify whether ExampleBank's services are enabled for a given user.

Table 9-4 ExampleBank's Service Enabled Attributes

Attribute Name	Attribute Description	Syntax	Multivalued Attribute
ebPhoneBankingEnabled	Specifies if phone banking services are enabled or not	Directory String	No
ebEBankingEnabled	Specifies if e-banking services are enabled or not	Directory String	No
ebPhoneBankingLoanServicesEnabled	Specifies if the Phone banking loan services are enabled or not	Directory String	No
ebEBankingLoanServicesEnabled	Specifies if the e-banking loan services are enabled or not	Directory String	No
ebEBankingCheckBalanceEnabled	Specifies whether the e-banking balance check service is enabled	Directory String	No
ebEBankingIntraBankTransfersEnabled	Specifies whether the e-banking intrabank transfer service is enabled	Directory String	No
ebEBankingInterBankTransfersEnabled	Specifies whether the e-banking inter-bank transfer service is enabled	Directory String	No
ebEBankingChangeProfileEnabled	Specifies whether the e-banking profile change service is enabled	Directory String	No
ebEBankingPortfolioManagement Enabled	Specifies whether the e-banking portfolio management service is enabled	Directory String	No
ebPhoneBankCheckBalanceEnabled	Specifies whether the phone banking balance check service is enabled	Directory String	No

Table 9-4 ExampleBank's Service Enabled Attributes *(Continued)*

ebPhoneBankIntraBankTransfers Enabled	Specifies whether the phone banking intrabank transfer service is enabled	Directory String	No
ebPhoneBankInterBankTransfers Enabled	Specifies whether the phone banking inter-bank transfer service is enabled	Directory String	No
ebPhoneBankChangeProfileEnabled	Specifies whether the phone banking profile change service is enabled	Directory String	No
ebPhoneBankPortfolioManagement Enabled	Specifies whether the phone banking portfolio management service is enabled	Directory String	No

Object Classes

The entries in the directory inherit from the `inetOrgPerson` object class and then the `ebPerson` object class. ExampleBank uses the object classes presented in the following tables:

Table 9-5 ebPerson Object Class

Object Class Name	`ebPerson`
Object Class Type	Structural
Superior Class	`inetOrgPerson`
Required Attributes	`ebid`
Allowed Attributes	`ebStatus, ebCurrentAccountNumber, ebSavingsAccountNumber, ebPreferredLanguage, ebSecondaryLanguage, ebCustomField1, ebCustomField2, ebCustomField3, ebCustomField4, ebAreaCode, ebNewsletterSubscription, ebNewsLetterType, ebCheckBalanceEnabled, ebEBankingEnabled, ebPhoneBankingEnabled`

Table 9-6 ebEBankingUser Object Class

Object Class Name	`ebEBankingUser`
Object Class Type	Auxiliary
Superior Class	`ebPerson`
Allowed Attributes	`ebEBankingCheckBalanceEnabled`, `ebEBankingIntraBankTransferEnabled`, `ebEBankingInterBankTransferEnabled`, `ebEBankingChangeProfileEnabled`, `ebEbankingPortfolioManagementEnabled`, `ebEBankingLoanServicesEnabled` in addition to additional attributes specific to services being provided.

Table 9-7 ebPhoneBankingUser Object Class

Object Class Name	`ebPhoneBankingUser`
Object Class Type	Auxiliary
Superior Class	`ebPerson`
Allowed Attributes	`ebPhoneBankingPin`, `ebPhoneBankCheckBalanceEnabled`, `ebPhoneBankIntraTransfersEnabled`, `ebPhoneBankInterTransfersEnabled`, `ebPhoneBankChangeProfileEnabled`, `ebPhoneBankPortfolioManagementEnabled`, `ebEBankingLoanServicesEnabled` in addition to additional attributes specific to services being provided.

Data

Based on the schema we have just examined sample container entries might look as follows:

```
dn: dc=eb,dc=com
objectclass:top
objectclass: organization
dc: eb
o:ExampleBank


dn: ou=people, dc=eb,dc=com
ou:people
description: Customers and employees of ExampleBank
objectclass:top
objectclass: organizationalunit
objectclass: example-am-managed-org-unit
```

Let us imagine that Bill Smith is an employee of ExampleBank and has both e-banking and phone banking services enabled. However, since his primary mode of access to his accounts is via the web, Bill Smith requested that he be able to check his balance and change his address over the phone, but that the other phone banking services be disabled. His sample entry would appear as follows:

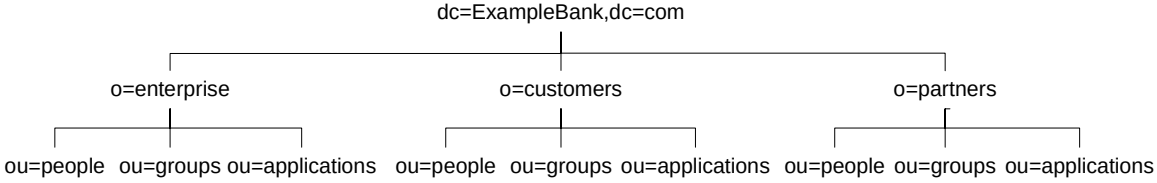
```
dn:ebid=123456789,ou=people,dc=eb,dc=com
objectclass:top
objectclass:person
objectclass:organizationalPerson
objectclass:inetOrgPerson
objectclass:ebPerson
objectclass:ebEBankingUser
objectclass:ebPhoneBankingUser
objectclass:example-am-web-agent-service
objectclass:example-am-managed-person
objectclass:example-am-user-device
objectclass:inetuser
objectclass:examplePreferences
objectclass:inetOrgPerson
objectclass: sunPortalDesktopPerson
objectclass: sunPortalNetmailPerson
ebid:123456789
displayname:Bill Smith
userPassword:{SSHA}Ek12JHYZ87op9645==
uid: 123456789
inetuserstatus: active
sn:Smith
```

```
givenname:William
cn:William F. Smith
mail:Bill.Smith@eb.com
telephonenumber:+1-256-556-5896
facsimiletelephonenumber:+1-256-556-5897
ebStatus:employee
ebAreaCode:CA-17B
ebPreferedLanguage:en
ebSecondaryLanguage:fr
ebCheckingsAccountNumber:133003300
ebSavingsAccountNumber:233003300
ebPhoneBankingEnabled:active
ebPhoneBankingPin:123456
ebEBankingEnabled:active
ebNewsletterSubscription:active
ebNewsletterType:email
ebEBankingCheckBalanceEnabled:active
ebEBankingIntraBankTransfersEnabled:active
ebEBankingInterBankTransfersEnabled:active
ebEBankingChangeProfileEnabled:active
ebEBankingPortfolioManagementEnabled:active
ebEBankingLoanServicesEnabled: inactive
ebPhoneBankCheckBalanceEnabled:active
ebPhoneBankingLoanServicesEnabled: inactive
ebPhoneBankIntraBankTransfersEnabled:inactive
ebPhoneBankInterBankTransfersEnabled:inactive
ebPhoneBankChangeProfileEnabled:active
ebPhoneBankPortfolioManagementEnabled:inactive
```

Directory Information Tree

In a desire to guard against the knock-on effect that the frequent reorganizations at ExampleBank could have on the hierarchical organization of its data, ExampleBank has decided to opt for a relatively flat directory information tree structure. This makes a clear distinction between employees, customers and partners by placing them into separate parts of the information tree, as indicated in Figure 9-3.

Figure 9-3 ExampleBank's Directory Information Tree



This distinction between customers, employees, and partners is preferred by ExampleBank for security reasons, as it allows them to implement a separate security policy for each branch. The distinction also provides more economical and easier searches, in addition to improved access control management, all of which translate into optimized performance and improved usability.

NOTE The directory information tree presented in Figure 9-3 reflects a design choice that presumes no prior directory structure. As many enterprises are likely to have a directory structure in place which does not include the `o=enterprise`, `o=customer`, and `o=partner` level, implementing such an information tree could involve non-negligible reworking of the directory structure and associated overheads. However, in terms of the potential subsequent performance and usability gains, this may prove to be a worthwhile investment.

In addition to the user management achieved through the directory information tree and groups mechanism, ExampleBank has chosen to implement the roles feature supported by Sun ONE Directory Server 5.2. The Directory Server roles functionality allows you to conveniently place users into meaningful sets of users,

and can not only be used internally by other Directory Server features such as access control, account lockout and password policy, but also by external applications such as, for example, a phone banking or human resources application that ExampleBank may implement.

Table 9-8 contains a list of roles that might be implemented by ExampleBank to facilitate user management. This is of course not an exhaustive list, but one which serves as a starting point for understanding how ExampleBank might consider grouping users involved in phone banking and e-banking services into roles to facilitate their management.

Table 9-8 Roles Implemented to Facilitate ExampleBank User Management

Role Name	Role Members	Role Characteristics
Customer Role	All ExampleBank Customers	Managed role that groups customers together.
Contractor Role	All ExampleBank Contractors	Managed role which groups together all contractors. This role limits access to certain sensitive employee and customer attributes.
Employee Role	All ExampleBank Employees	Managed role that groups employees together .
Phone Banking Team Role	All Phone Banking team members involved in running the phone banking service	Managed role which groups together all members of the phone banking team and grants access to phone banking attributes in users' entries except the `ebPhoneBankingLoanServicesEnabled` attribute.
Trusted Contributor Role	All ExampleBank employees and certain contractors requiring access to employee phone book information.	Nested role which extends the scope of the Employee Role to include those contractors requiring access to employee phone book data. Role members have read, search and compare access to employee phone book data.
e-banking Team Role	All e-banking team members involved in running the e-banking service	Managed role which groups together all members of the e-banking team and grants access to the e-banking attributes in users' entries except the `ebEBankingLoanServicesEnabled` attribute.
Phone Banking Loan Manager Role	All Phone Banking Managers entitled to take loan allocation decisions.	Managed role which groups together all senior managers in the Phone Banking Team and grants access to ALL phone banking attributes including the `ebPhoneBankingLoanServicesEnabled` attribute.

Table 9-8 Roles Implemented to Facilitate ExampleBank User Management *(Continued)*

e-banking Loan Manager Role	All e-banking Managers entitled to take loan allocation decisions.	Managed role which groups together all senior managers in the e-Banking Team and grants access to ALL e-banking attributes including the `ebEBankingLoanServicesEnabled` attribute.
Loan Manager Role	All Phone Banking and e-banking Managers entitled to take loan allocation decisions.	Nested role which unifies the Phone Banking Loan Manager Role and the e-banking Loan Manager Role. Role members have access to the `ebPhoneBankingLoanServicesEnabled` and `ebEBankingLoanServicesEnabled` attributes.

The roles listed in Table 9-8 are located under the `ou=people` subtree of the `o=enterprise`, `o=customer` or `o=partner` branches of the directory information tree depending on where they belong. They are managed by a Role Manager entry, which has access to both the roles and the attributes within those roles. This Role Manager entry is located under `o=enterprise`. Access controls are established to govern the access rights of each role, and the Role Manager.

Code Example 9-1 shows what the Trusted Contributor role entry would look like in an ldif file, providing a more detailed view of how this role functionality might be implemented.

Code Example 9-1 ldif for Trusted Contributor Role Entry

```
dn:cn=TrustedContributorRole,ou=people,o=enterprise,
dc=ExampleBank,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsNestedRoleDefinition
cn: TrustedContributorRole
nsRoleScopeDN: o=partners,dc=ExampleBank,dc=com
nsRoleDN: cn=EmployeeRole,o=enterprise,dc=ExampleBank,dc=com
nsRoleDN: cn=ContractorRole,o=partners,dc=ExampleBank,dc=com
```

The access control granting all members of this Trusted Contributor role read, search and compare access on phone book data, would appear under `o=enterprise,dc=ExampleBank,dc=com` as shown in Code Example 9-2:

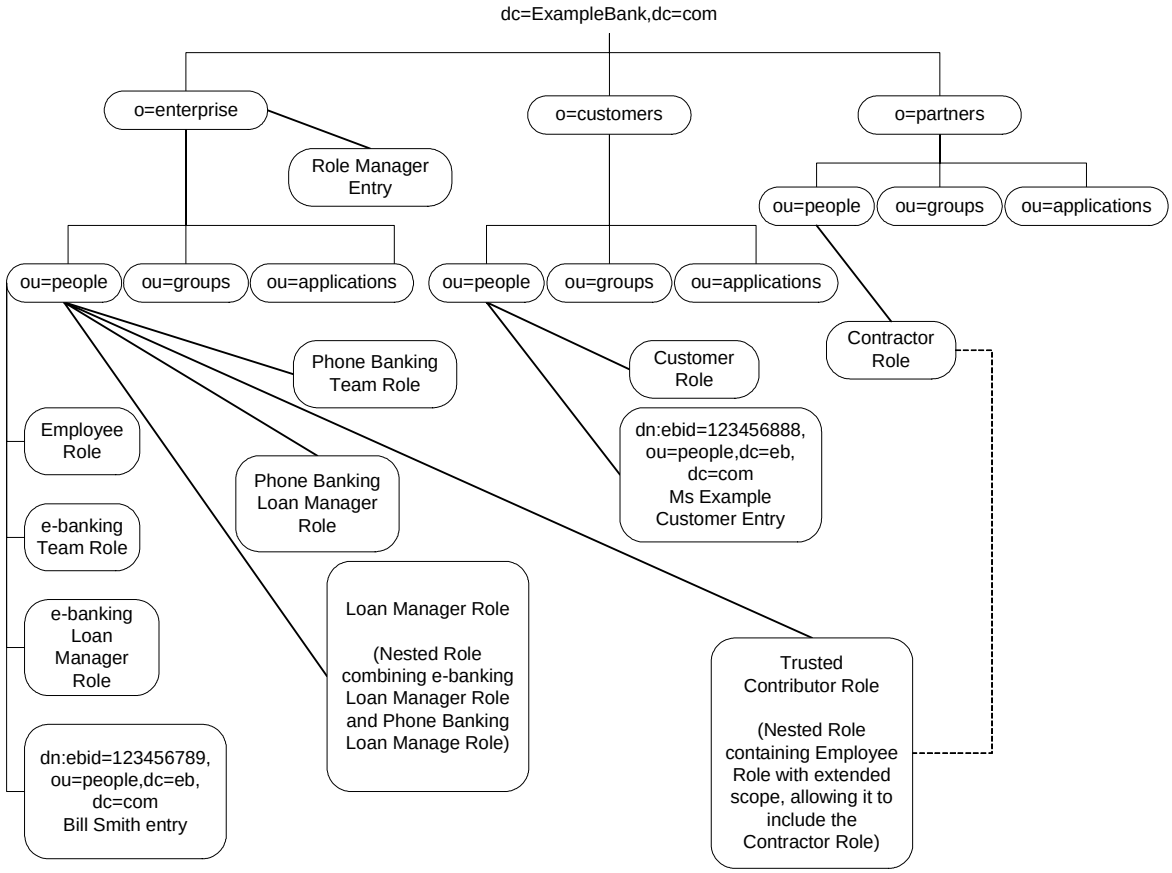
Code Example 9-2 Access Control Granting Read, Search, and Compare Rights to Phone Book Data to all Trusted Contributor Role Members

```
aci:(targetattr="telephoneNumber || mail  ||facsimileTelephonenumber")(version
 3.0; aci "authorize for search,read,compare";allow(search,read,compare)
 roledn = "ldap:///cn=TrustedContributorRole,ou=people,o=enterprise,
 dc=ExampleBank,dc=com";)
```

NOTE As this extended scope for roles functionality is new to Directory Server 5.2, Sun ONE Identity Server 6.0 would not recognize this new functionality.

Figure 9-4 on page 257 provides a global view of the role configuration and how it fits into the directory information tree structure:

Figure 9-4 ExampleBank's Roles, Role Manager and Sample Entries



Security Considerations

In a financial environment providing uncompromised security is crucial to success. With Directory Server as part of its online banking deployment stack, ExampleBank can offer optimum security in terms of protecting channels, governing access to data, encrypting sensitive data while in storage, providing flexible password policies, and optimizing SSL connections.

This section describes ExampleBank's security policies and is divided into the following subsections:

- Securing Communication Channels
- Securing Data in Storage
- Securing Password Authentication

Securing Communication Channels

A first imperative for ExampleBank is to ensure that all communication channels between the different elements in the deployment stack are secured. To this end, ExampleBank implements SSL, which ensures the transport security. Furthermore, to enhance the performance for connections using the Secure Sockets Layer (SSL) protocol with certificate-based authentication, ExampleBank takes advantage of the new Sun Crypto Accelerator Board functionality provided with Directory Server 5.2. For information on how to install and configure the Sun Crypto Accelerator Board see Appendix B "Using the Sun Crypto Accelerator Board" in the *Sun ONE Directory Server Installation and Tuning Guide.*

Securing Data in Storage

A second imperative is to provide maximum protection for sensitive data, such as pin numbers, when in storage. ExampleBank uses Sun ONE Directory Server 5.2's attribute encryption feature to protect data. This attribute encryption feature allows ExampleBank to specify that certain attributes be stored in an encrypted form. This feature is configured at database level, which means that once ExampleBank decides it wants to encrypt an attribute, that particular attribute will be encrypted for every entry in the database. When in storage, encrypted attributes are prefaced with a cipher tag which indicates the encryption algorithm used. An encrypted attribute using the DES encryption algorithm would appear as follows:

```
{DES}3hakc&jla+=snda%
```

It is important to realize from a security standpoint, that because the actual encryption is at attribute level, rather than entry level, the only way to encrypt an entire entry would be for ExampleBank to encrypt all of its attributes. It is also important to understand that because the aim is to protect the sensitive data when it is in storage, attribute encryption is always reversible, i.e encrypted attributes are decrypted when returned via search requests, hence the need to secure communication channels with SSL.

Sun ONE Directory Server 5.2's attribute encryption feature provides ExampleBank with the levels of security it requires in that the feature uses the private key of the server's SSL certificate, to generate its own key, which is then used to perform the encryption and decryption operations. ExampleBank is therefore obliged to have an SSL configuration in place which provides the generated key before being able to proceed with encryption, and more importantly decryption operations. What is more ExampleBank benefits from the fact that the Directory Server attribute encryption feature is based on the NSS library, in that it can choose from different encryption algorithms and have guaranteed portability across different platforms.

Securing Password Authentication

ExampleBank has a user population of employees, customers, contractors and business partners, all of whom are authenticating to the online banking stack via the third party single sign-on application. The desire at ExampleBank is to impose more stringent password policies, on certain users such as contractors, and have less stringent password policies governing customers and employees. This desire can be fulfilled due to the fact that the third-party single sign-on application can take advantage of Sun ONE Directory Server 5.2's provision for multiple password policies. Individual password policies can be defined for either a given user or a set of users. A natural way of assigning a password policy to a set of users is to configure the CoS definition to provide values for the `passwordPolicySubentry` attribute in user entries as a function of the roles that those user entries have.

So, not only does ExampleBank have the ability to tailor password policies to user security requirements, but it can use the roles which are already defined, to make for easier password policy management.

Implementation

Rolling out the Sun ONE Directory Server Online banking stack is a large undertaking, and one which requires extensive planning, analysis, and organizational support.

NOTE We cannot stress enough the need for tight coordination between all parties involved, whether they be marketing team players who generate user profile needs information or system designers. This tight coordination, which will allow you to draw up a common architecture strategy and benefit from a truly centralized decision making structure, is the key to the success of your implementation. To avoid recreating a disparate status quo, this tight coordination MUST remain the top priority throughout the entire operation.

Adequate training and documentation on both the Sun ONE support staff side and the customer side help ensure that deliveries are consistent and constitute the backbone of this tight coordination.

To gain a comprehensive overview of creating and implementing a directory pilot before you begin your implementation, we highly recommend you refer to *Understanding and Deploying LDAP Directory Services* (T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999).

The complexity of the implementation requires a phased approach, an outline of which follows. You will need to stagger your implementation over time into logical phases, which are likely to resemble the following:

- Analyze and Plan Your Directory Infrastructure

  The functional and business requirements of the Directory Server Infrastructure are assessed and analyzed at this stage. This analysis helps identify any functional gaps that may require system customizing or extension. A technical review of the production environment of the existing implementation will be conducted to determine potential issues with scalability, performance and reliability of the proposed hardware and network environment.

  High-level architecture definition follows on from the above analysis and it is at this stage that the recommendations for sizing, scaling, performance, physical distribution, replication and referrals, security, failover, backup, and synchronization with other data sources and authentication mechanisms are drawn up.

- Construct and Design Directory Infrastructure

  During this second phase, Sun ONE together with the ExampleBank project team architect, design and construct the core online banking deployment stack. The key areas of activity at this stage include determining server sizing and placements, server configurations, integrating existing backend applications into the directory infrastructure, planning deployment and administration processes and identifying any remaining functionality gaps.

- Implement Core Directory Infrastructure

  Sun ONE implements and deploys the core Directory Infrastructure to production during this phase. It is essential that appropriate technology guidance and mentoring in server configuration and administration are made available. Together with Sun ONE the project team assesses the impact on end users and draws up the necessary communication and training plans.

- Enable Enterprise to Employee (E2E) Functionality

  The first area of functionality to be enabled is enterprise to employee (E2E) functionality and typically includes the following tasks in the preliminary stages:

  - Provide an LDAP-accessible list of users
  - Map authoritative sources and clean up user data
  - Automatically generate standard IDs
  - Flag terminated accounts
  - Enable LDAP applications

- Authenticate with intranet applications

Once the above are operational, the finer details of the E2E functionality can be addressed. Linkages between the different data repositories are automated, incremental expansion takes place and any security requirements such as password synchronization are resolved.

The penultimate stage concentrates on providing a scalable single sign-on architecture for all the web and back-end applications. Implementation testing follows and must include integration, performance, negative and user acceptance testing of the E2E functionality. Once again end user impact in relation to the single sign-on architecture will be assessed and provided for in terms of appropriate training and communication.

Finally, once the Directory Server Infrastructure is enabled to provide single sign-on for the platform's web and back-end applications, the project team will document the procedures to assist implementing similar E2E functionality for other applications and business groups within the company.

- Expand Core Directory Server Infrastructure to Support Additional Common Services

  The goal is to continue to expand the support additional common services and functionality for further E2E, Business-to-Business (B2B), and / or Business-to-Consumer (B2C) requirements. Specific B2B and B2C functionality will be defined, designed and implemented as and when the business units' requirements are determined and approved.

As you can see rolling out a Directory Server deployment stack is no light affair, and we strongly recommend the involvement of Sun Professional Services, to optimize your implementation. Contact Sun ONE Professional Services at: `http://www.sun.com/service/sunps/sunone/`

Chapter 10

Architectural Strategies

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 ONE 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 and performance 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 *Sun ONE 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

Sun ONE Directory Server provides two methods of backing up data: binary backup (`db2bak`) and backup to an ldif file (`db2ldif`). 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 the Sun ONE 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 *Sun ONE 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).

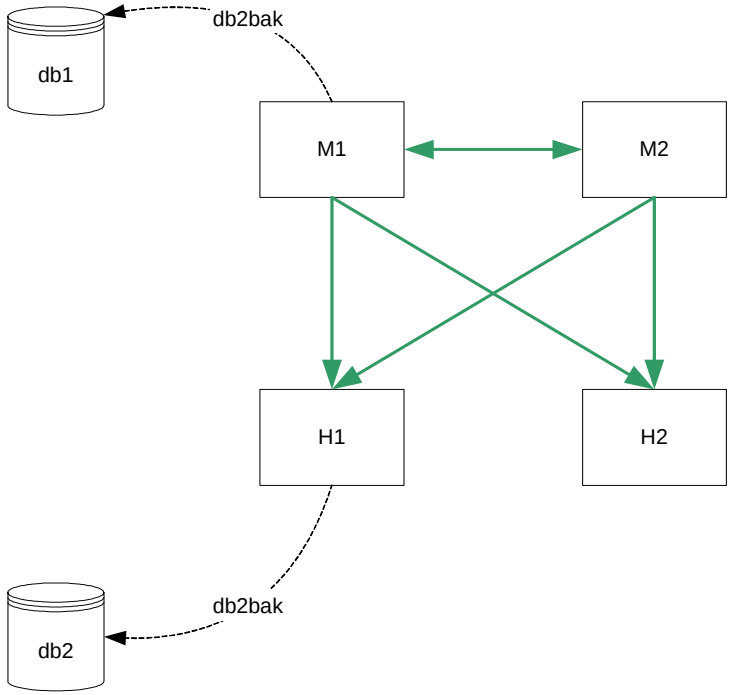
NOTE Because it is easier to restore from a local backup, it is recommended that you perform a binary backup on each server.

In all of the diagrams that follow in this chapter:

- M = master
- H = hub
- C = consumer
- RA = replication agreement.

Figure 10-1 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 10-1 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.)

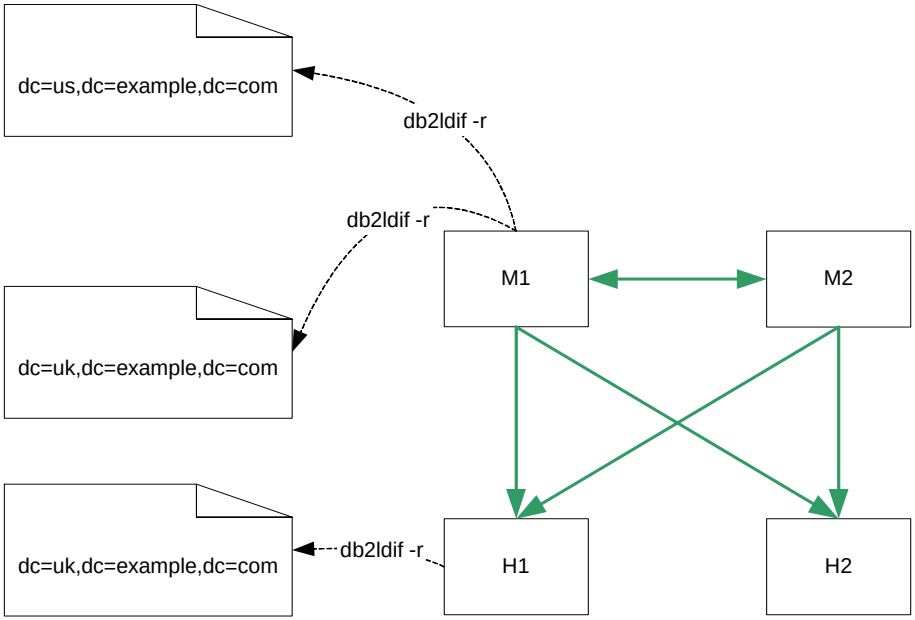
Backing up using backup to ldif has the following limitations:

- In situations where rapid backup and restoration are required, backup to ldif may take too long to be viable.

It is recommended that you 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 10-2 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

Sun ONE 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.

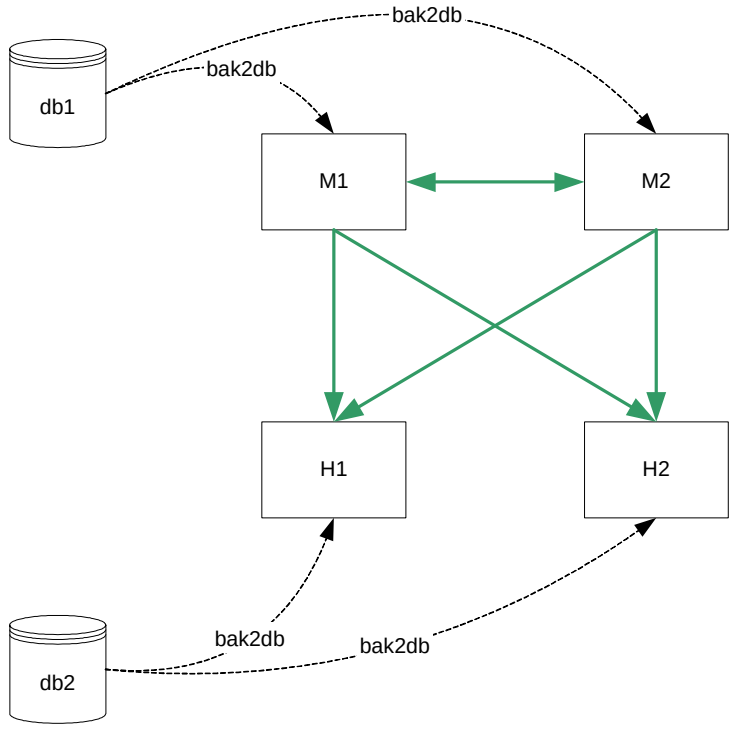
Restoring data using binary restore has the following limitations:

- Restoration can be performed only on a server with an identical configuration, as defined on page 265. For more information on restoring data with the binary restore feature, refer to Initializing a Replica Using Binary Copy in the *Sun ONE 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 recommended restoration method if the machines have an identical configuration, and time is a major consideration.

Figure 10-3 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 10-3 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:

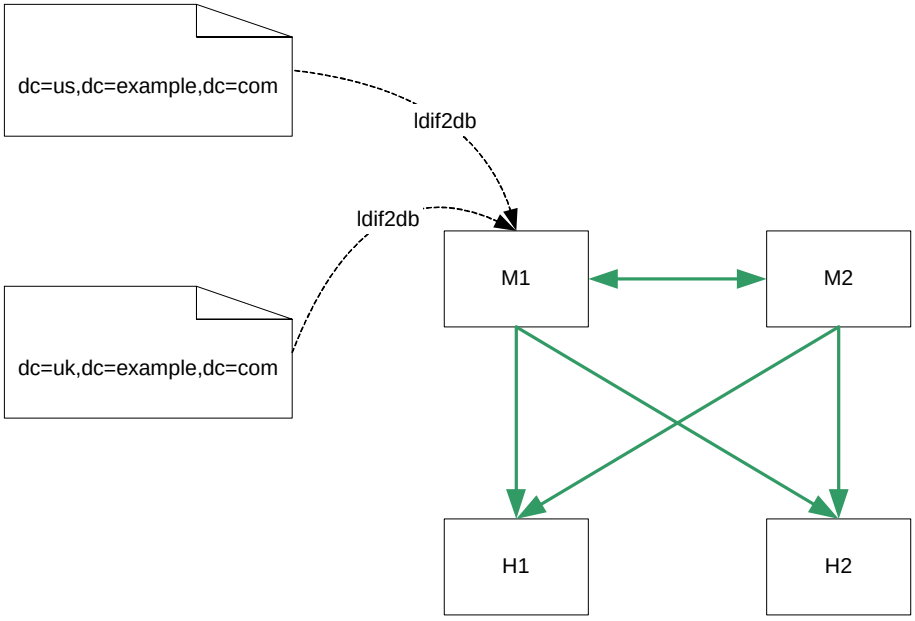
- It can be performed on any server, regardless of its configuration.
- A single ldif file can be used to deploy an entire directory service, regardless of its replication topology. This is particularly useful for the dynamic expansion and contraction of a directory service according to anticipated business 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 10-4 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.

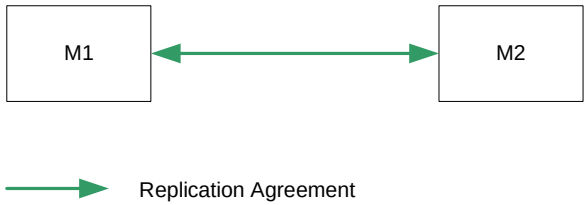
Single Data Center

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 10-5 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.

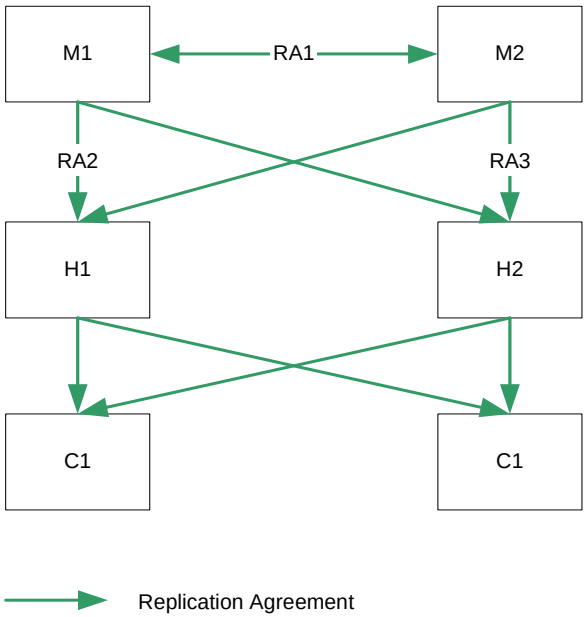
Figure 10-5 One Data Center - Basic Topology



Single Data Center Scaled For Read Performance

Increased read performance is achieved by adding hubs, and then consumers, as indicated in Figure 10-6. 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 10-6 One Data Center Scaled For Read Performance



Single Data Center Failure Matrix

In the scenario depicted in Figure 10-6, various components may be rendered unavailable for any one of the reasons described on page 264. These points of failure, and the related recovery actions are described in table Table 10-1.

Table 10-1 Single Data Center - Failure Matrix

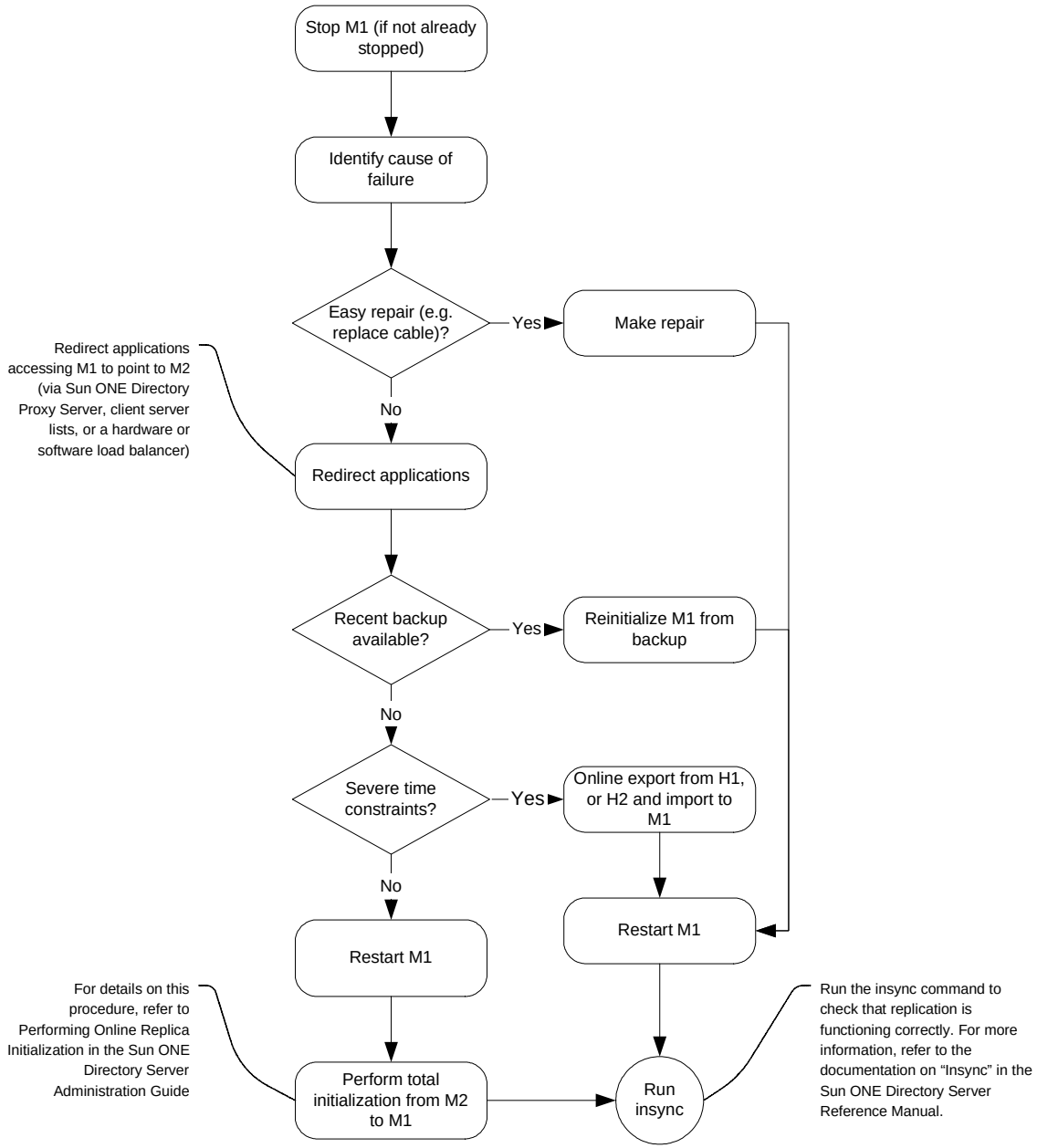
Failed Component	Action
M1	Local writes are routed to M2, via Sun ONE Directory Proxy Server, client server lists, or a hardware or software load balancer. M2 continues to replicate to both H1 and H2.
M2	Local writes are routed to M1, via Sun ONE Directory Proxy Server, client server lists, or a hardware 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 10-7, and the procedure that follows, assumes that one master (M1) has failed.

Figure 10-7 Single Data Center Recovery Sample Procedure (One Component)



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 ONE 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 the *Sun ONE 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 the *Sun ONE Directory Server Reference Manual*.

NOTE Performing an online export will impact the performance of the server. It is therefore recommended that you 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 the *Sun ONE 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

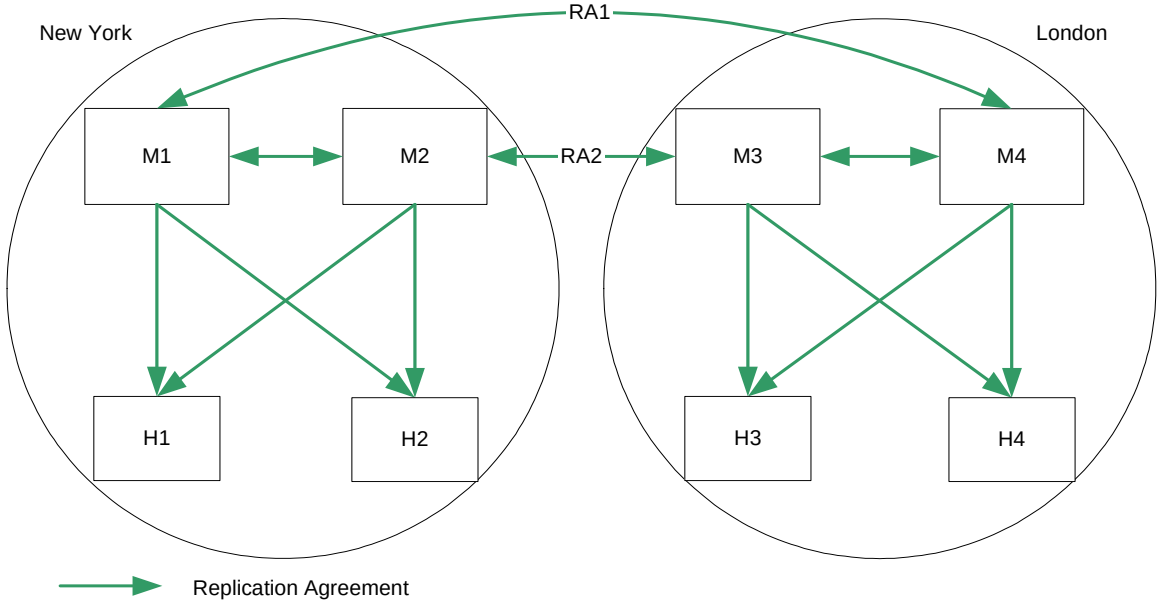
When data is shared across sites, an effective replication topology is imperative, for both performance and failover.

Two Data Centers Basic Topology

The topology depicted in Figure 10-8 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, it is recommended that 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 10-8 Two Data Centers Basic Topology

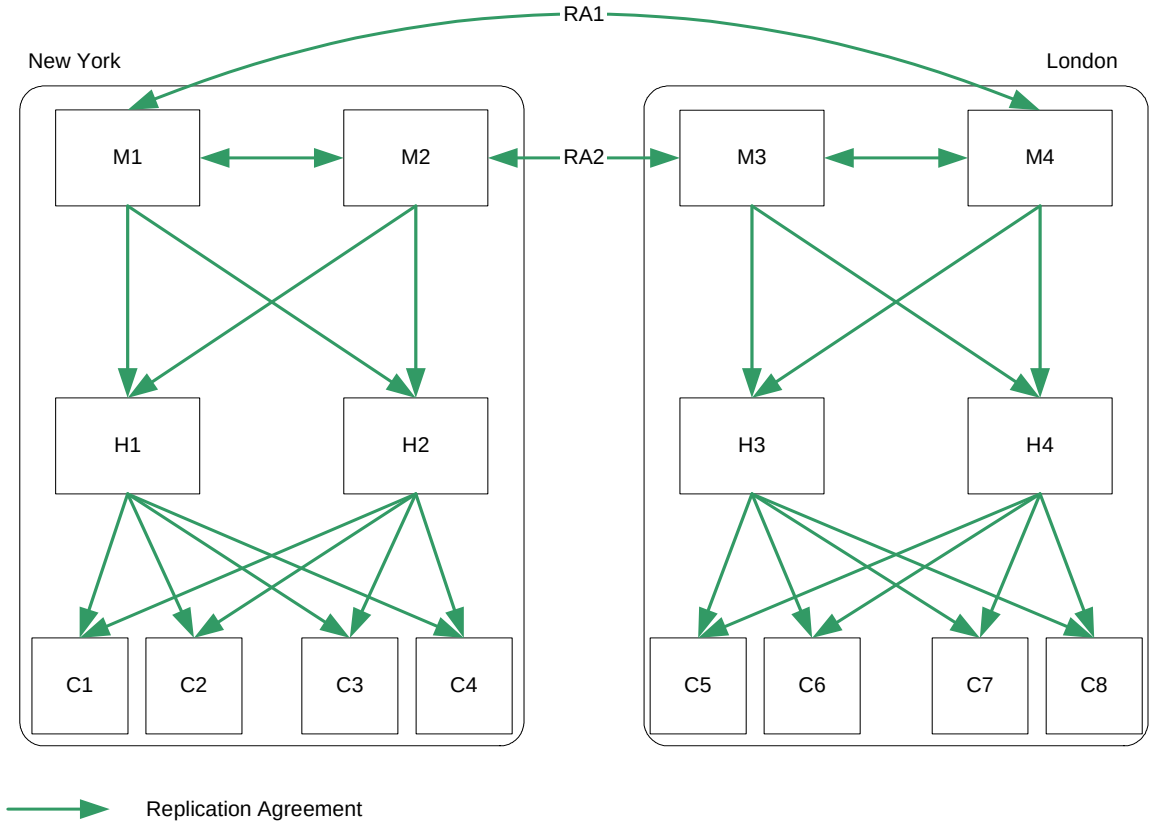


Two Data Centers Scaled For Read Performance

As in the scenario for one data center, increased read performance is achieved by adding hubs, and then consumers, as indicated in Figure 10-6.

In this scenario, it is recommended that 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 10-9 Two Data Centers Scaled For Read Performance



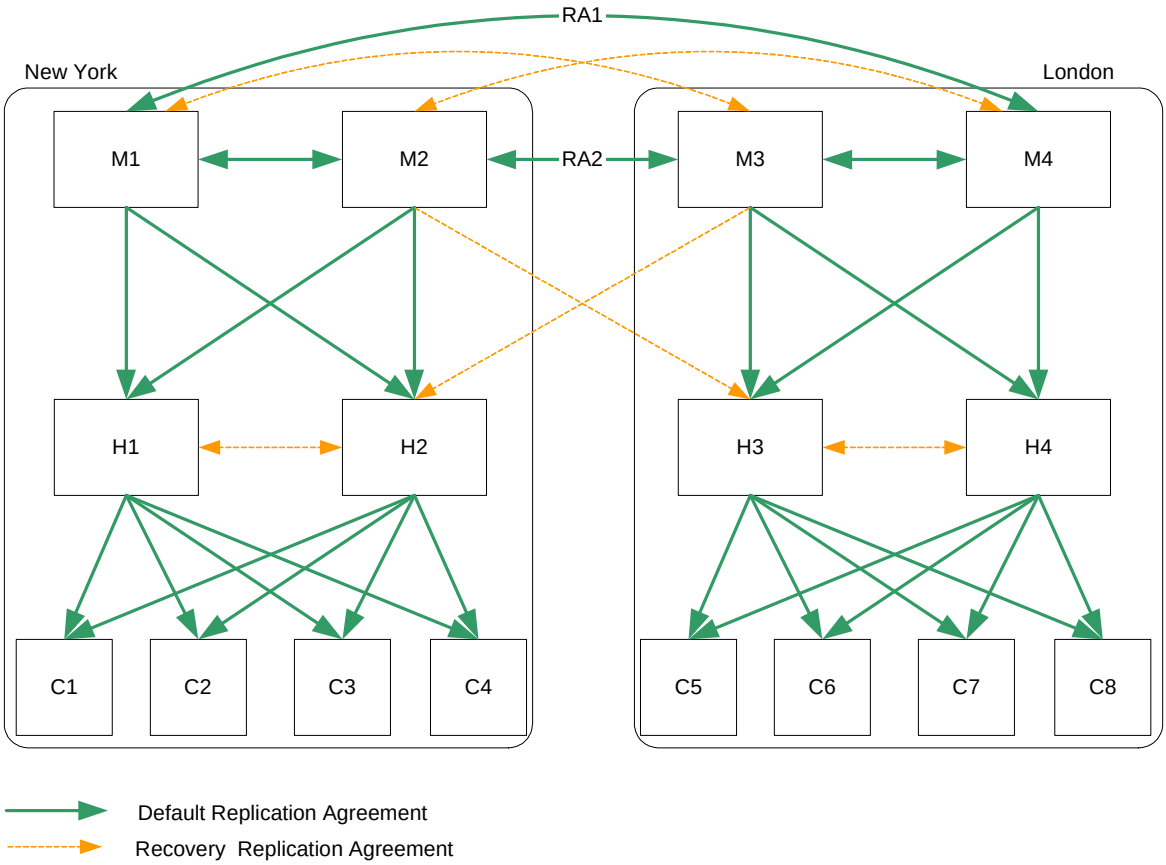
Two Data Centers Recovery Scenarios

For the deployment depicted in Figure 10-9, 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 10-10.

Figure 10-10 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 10-10, 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 the *Sun ONE 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.

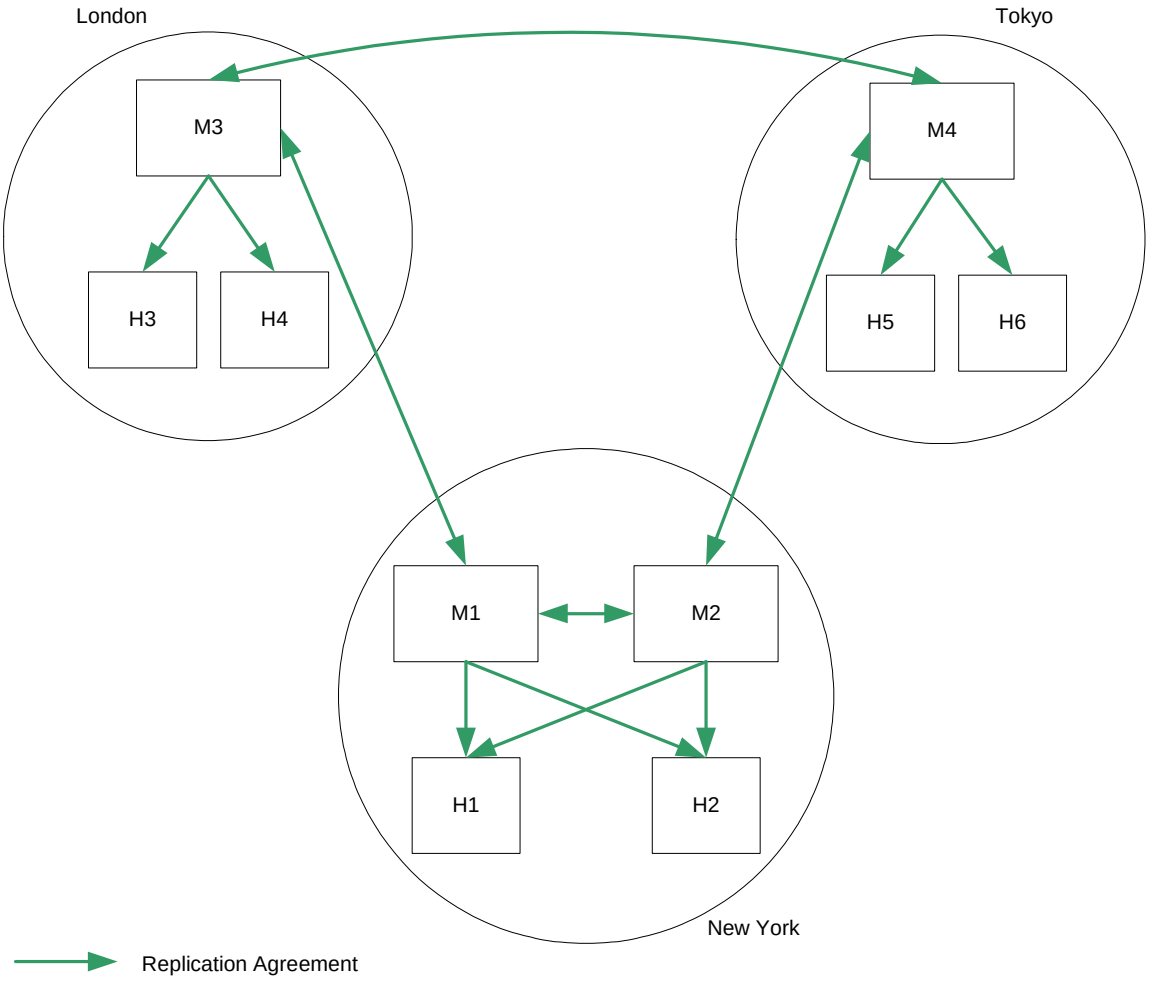
Three Data Centers

Directory Server 5.2 supports 4-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 10-11 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.

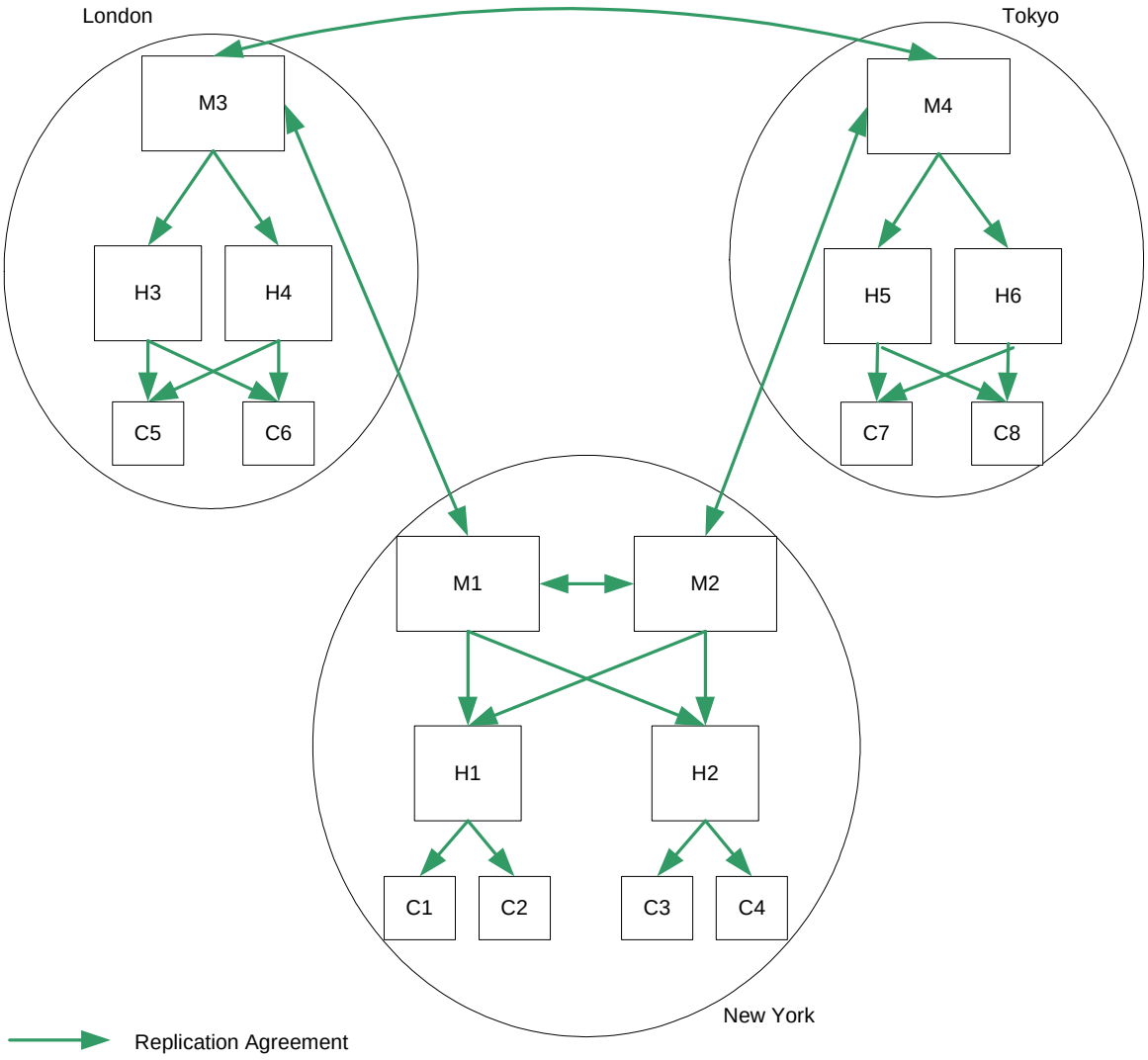
Figure 10-11 Three Data Centers Basic Topology



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. The recommended topology is indicated in Figure 10-12.

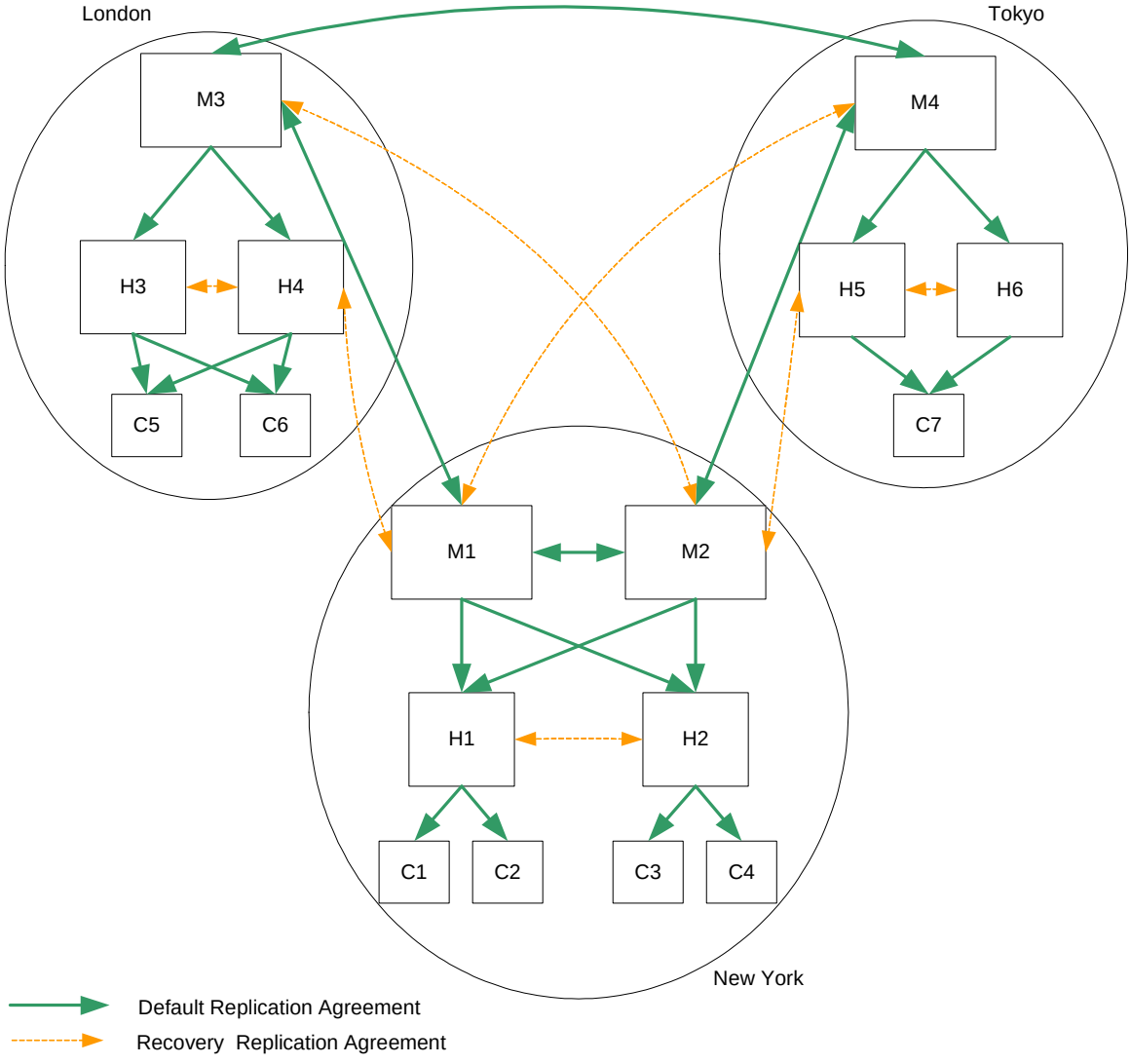
Figure 10-12 Three Data Centers Scaled For Read Performance



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 10-13.

Figure 10-13 Three Data Centers Recovery Replication Agreements



Three Data Centers Recovery Procedure (One Component)

In the scenario depicted in Figure 10-13, 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 the *Sun ONE 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.

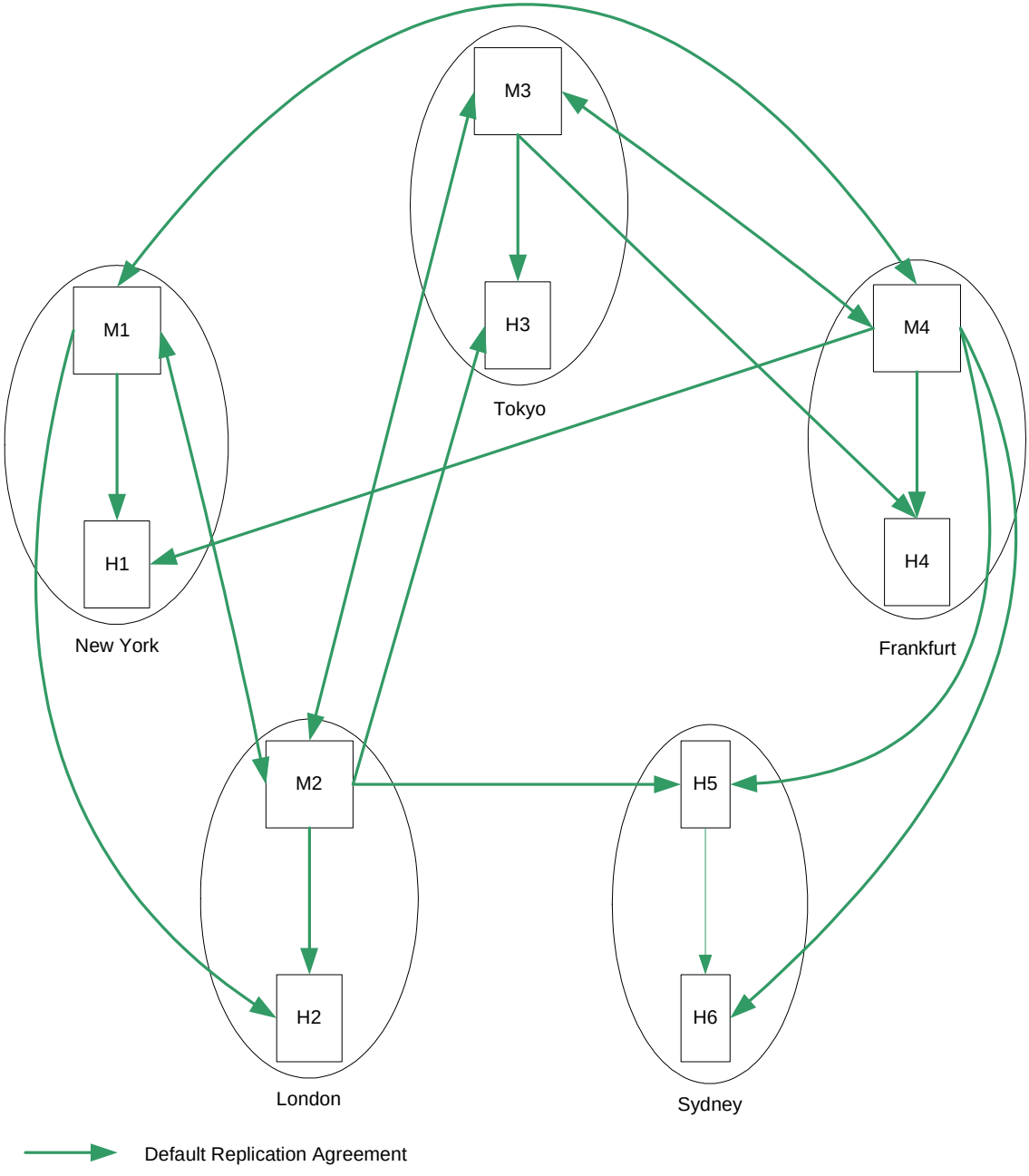
Five Data Centers

Sun ONE Directory Server 5.2 supports 4-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 10-14 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 10-14 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 10-15 on page 289.

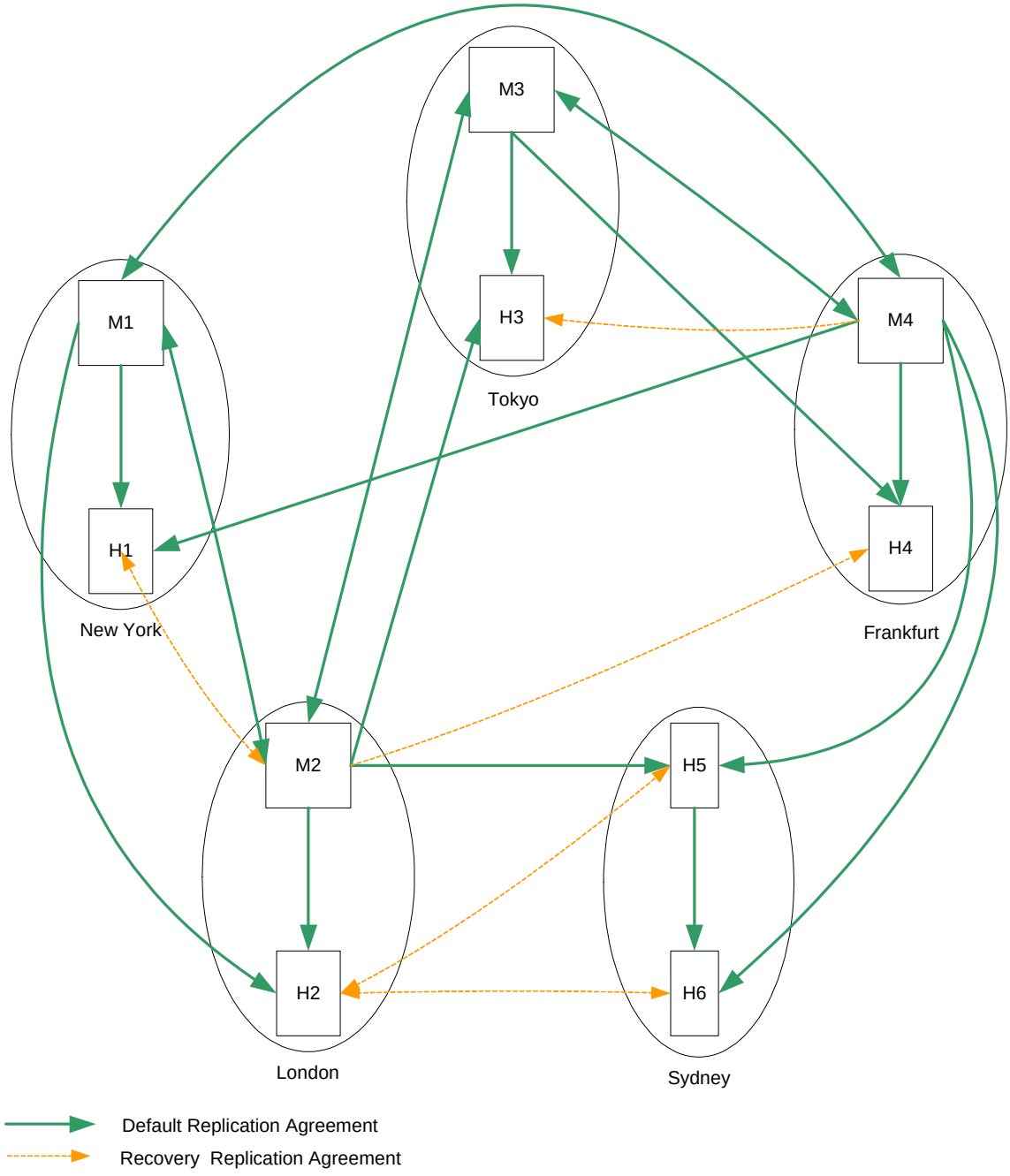
Five Data Centers Recovery Procedure (One Component)

In the scenario depicted in Figure 10-15, 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 the *Sun ONE 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 10-15 Five Data Centers Recovery Replication Agreements



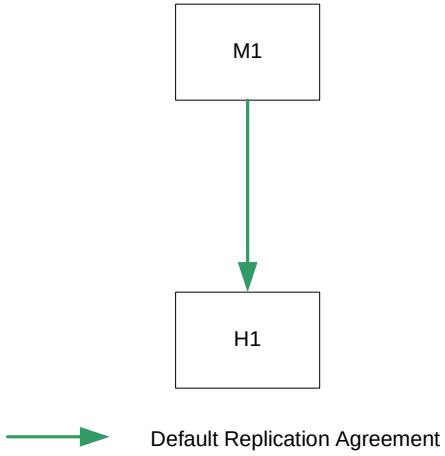
Single Data Center Using the Retro Change Log Plug-In

The previous topology for a single data center does not take into account the use of the retro change log plug-in. Compatibility with this plug-in implies that a multi-master replication topology can not be deployed. For more information on the retro change log plug-in, refer to Using the Retro Change Log Plug-In in the *Sun ONE Directory Server Administration Guide*.

Retro Change Log Plug-in Basic Topology

If multi-master replication cannot be deployed, the basic topology depicted in Figure 10-16 is suggested.

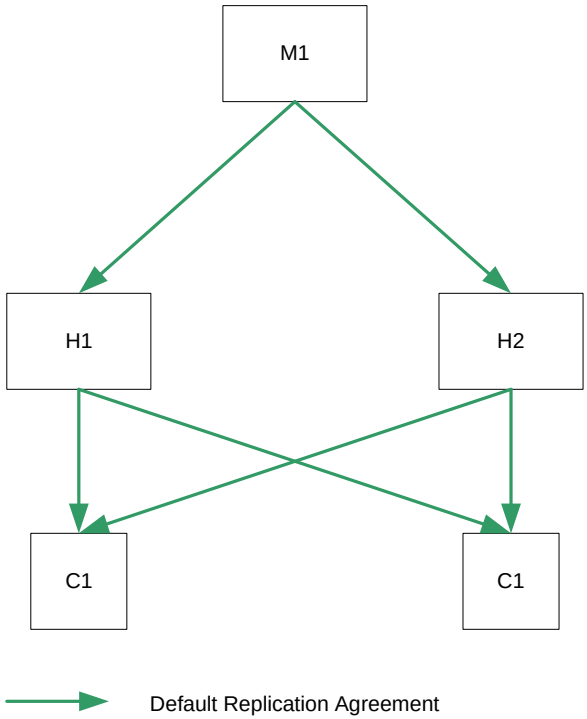
Figure 10-16 One Data Center Using the Retro Change Log Plug-in



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 10-17.

Figure 10-17 One Data Center Using the Retro Change Log Plug-in (Scaled)



Retro Change Log Plug-in Recovery Procedure

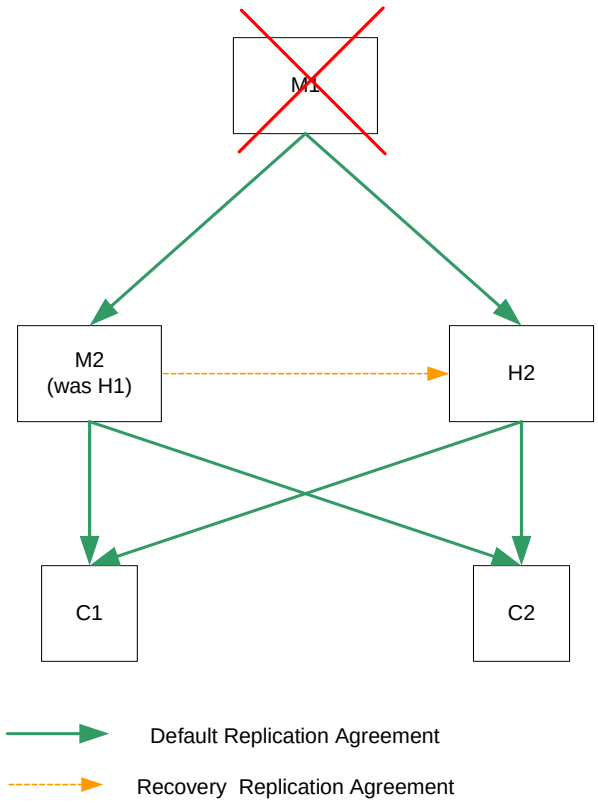
For the deployment depicted in Figure 10-17, the following strategy can be applied if the master server fails:

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 the *Sun ONE 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 10-18.

Figure 10-18 One Data Center Using the Retro Change Log Plug-in (Recovery)



Appendix A

Accessing Data Using DSMLv2 Over HTTP/SOAP

Sun ONE Directory Server 5.2 supports DSMLv2, which maps the majority of the query/update functionality of LDAP version 3 to XML, SOAP and HTTP. Being able to perform directory operations in DSMLv2 extends the reach of your directory data directly into the worlds of XML and web services. This section is designed to give you an insight into how you can perform directory operations using DSMLv2 over SOAP/HTTP and contains the following examples:

- An Empty Anonymous DSML "Ping" Request
- A DSML Request Issuing a User Binding
- A DSML Search Request

CAUTION The `content-length:` header in the following examples contains the exact length of the DSMLv2 request. In order for these examples to function correctly, make sure either that the editor you use respects these content lengths or that you modify them accordingly.

For a complete list of DSML related attributes and information about the DSMLv2 standard, refer to the *Sun ONE Directory Server Reference Manual*. For DSML related procedures refer to the *Sun ONE Directory Server Administration Guide*.

An Empty Anonymous DSML "Ping" Request

Before issuing DSML requests over HTTP/SOAP you need to ensure that the DSML front end is enabled by sending an empty DSML batch request to your directory. An empty DSML batch request would read as follows:

```
POST /dsml HTTP/1.1
content-length: 451
HOST: hostMachine
SOAPAction: ""
Content-Type: text/xml
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>

   <soap-env:Body>
      <batchRequest
          xmlns='urn:oasis:names:tc:DSML:2:0:core'
          requestID='Ping!'>
          <!-- empty batch request -->
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

The first section of this DSML request is the HTTP section:

```
POST /dsml HTTP/1.1
content-length: 451
HOST: hostMachine
SOAPAction: ""
Content-Type: text/xml
Connection: close
```

and is comprised of the HTTP method line

`POST /dsml HTTP/1.1`

followed by a number of HTTP headers. The HTTP method line specifies the HTTP method request and URL to be used by the DSML front end. `POST` is the only HTTP method request accepted by the DSML front end, whereas the `/dsml` URL, although the default URL for Directory Server, can be configured with any other valid URL. The HTTP headers that follow, specify the remaining details of the DSML request. The header

`content-length: 451`

gives the exact length of the SOAP / DSML request, and the header

`HOST: hostMachine`

specifies the name of the host Directory Server being contacted. The header

`SOAPAction:`

is mandatory and informs the directory that you want to perform a DSML request on the HTTP / SOAP stack. It may however, be left empty. The header

`Content-Type: text/xml`

must have a value of `text/xml` which defines the content as XML . Finally the header

`Connection: close`

specifies that the connection will be closed once the request has been satisfied, in contrast to the default HTTP / 1.1 behavior that maintains connections open.

The following constitutes the SOAP / DSML section of the request:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>

   <soap-env:Body>
      <batchRequest
          xmlns='urn:oasis:names:tc:DSML:2:0:core'
          requestID='Ping!'>
          <!-- empty batch request -->
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

The DSML request begins with the XML prolog header,

```
<?xml version='1.0' encoding='UTF-8'?>
```

which specifies that the request must be encoded with the UTF-8 character set, and then continues with the SOAP envelope and body element that contain the mandatory inclusion of the XML schema, XML schema instance and SOAP namespaces as follows:

```
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>

   <soap-env:Body>
```

The DSML batch request element

```
      <batchRequest
```

marks the beginning of the DSML batch request, is immediately followed by the mandatory inclusion of the DSMLv2 namespace,

```
xmlns='urn:oasis:names:tc:DSML:2:0:core'.
```

and is optionally identified by the following request ID

```
requestID='Ping!'>
```

The empty batch request

`<!-- empty batch request -->`.
is XML commented as such, and the SOAP / DSML batch request is closed using the following close batch request, close SOAP body, and close SOAP envelope elements:

```
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

CAUTION Maximum limits exist for the number of clients connecting simultaneously to the directory and for the size of the DSML requests. The limit for the number of clients is managed by the `ds-dsml-poolsize` and `ds-dsml-poolmaxsize` attributes and the request size limit by the `ds-dsmll-requestmaxsize` attribute. For more information regarding DSML-related attributes, refer to the *Sun ONE Directory Server Reference Manual.*

If the DSML front end is enabled, an empty DSML response, such as the one below will be returned.

```
HTTP/1.1 200 OK
Cache-control: no-cache
Connection: close
Date: Mon, 09 Sep 2002 13:56:49 GMT
Accept-Ranges: none
Server: Sun-ONE-Directory/5.2
Content-Type: text/xml; charset="utf-8"
Content-Length: 500

<?xml version='1.0' encoding='UTF-8' ?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'
   >
<soap-env:Body>
<batchResponse
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns='urn:oasis:names:tc:DSML:2:0:core'
   requestID='Ping!'
   >
</batchResponse>
</soap-env:Body>
</soap-env:Envelope>
```

However, if nothing is returned, then you can conclude that the front end is disabled.

NOTE The DSML front end is *disabled* by default. To enable it you should change the value of the `nsslapd-pluginEnabled` attribute under the `cn=DSMLv2-SOAP-HTTP,cn=frontends,cn=plugins,cn=config` entry in the `dse.ldif` file to on. You can also modify the `ds-hdsml-port` and `ds-hdsml-root` attributes, but this is optional. For information on how to enable the DSML front end, refer to the *Sun ONE Directory Server Administration Guide*.

A DSML Request Issuing a User Binding

To issue a DSML request you can choose, as with the LDAP protocol, either to bind to the directory as a given user or access anonymously. If you want to bind to the directory as a given user, your DSML request will include an HTTP authorization header containing a uid and a password that will then be mapped to a dn.

NOTE For information on how to configure the identity mapping refer to the *Sun ONE Directory Server Administration Guide*.

An HTTP authorization request would read as follows:

```
POST /dsml HTTP/1.1
content-length: 578
Content-Type: text/xml; charset="utf-8"
HOST: hostMachine
Authorization: Basic ZWFzdGVyOmVnZw==
SOAPAction: ""
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>
   <soap-env:Body>
     <batchRequest
        xmlns='urn:oasis:names:tc:DSML:2:0:core'
        <extendedRequest>
          <requestName>1.3.6.1.4.1.4203.1.11.3</requestName>
        </extendedRequest>
     </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

This example shows the HTTP authorization header transporting the uid easter and the password egg, which, in clear, would appear as `easter:egg`, and encoded in base64 as:

```
Authorization: Basic ZWFzdGVyOmVnZw==
```

Should you want to access anonymously, no HTTP authorization header including the password and uid would be required. However, it is worth bearing in mind that anonymous access is often subject to strict access controls, and that you would be likely to encounter data access restrictions. Similarly, you can issue DSML requests to perform LDAP operations by LDAP proxy. It is interesting to note that as DSML requests are managed on a batch basis, if you decide to issue requests by LDAP proxy, then the required DSML proxy authorization request must be the first in a given batch of requests.

A DSML Search Request

Once you have established that your DSML front end is enabled you can start performing directory operations. A sample DSML base object search request on the root DSE entry for the `namingContexts`, `supportedLDAPversion`, `vendorName`, `vendorVersion`, and `supportedSASLMechanisms` attributes would read as follows:

```
POST /dsml HTTP/1.1
HOST: hostMachine
Content-Length: 1081
Content-Type: text/xml
SOAPAction: ""
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'
   >
   <soap-env:Body>
      <batchRequest
        xmlns='urn:oasis:names:tc:DSML:2:0:core'
        requestID='Batch of search requests'
        >
        <searchRequest
            dn=""
            requestID="search on Root DSE"
            scope="baseObject"
            derefAliases="neverDerefAliases"
            typesOnly="false"
            >
            <filter>
               <present name="objectClass"/>
            </filter>
            <attributes>
               <attribute name="namingContexts"/>
               <attribute name="supportedLDAPversion"/>
               <attribute name="vendorName"/>
               <attribute name="vendorVersion"/>
               <attribute name="supportedSASLMechanisms"/>
            </attributes>
        </searchRequest>
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

With the following attribute-data pairs:

```
dn=""
requestID="search on Root DSE"
```

this search operation requests data under the root DSE entry and is identified with an optional request ID attribute. The attribute-data pair:

```
scope="baseObject"
```

specifies that the search is a base object search while the attribute-data pair:

```
derefAliases="neverDerefAliases"
```

specifies that the aliases should not be dereferenced while searching or locating the base object of the search, which is in fact the only `derefAliases` value supported by Directory Server.

The attribute-data pair:

```
typesOnly="false"
```

specifies that both the attribute names and their values be returned, as opposed to `typesOnly="true"`, which will return attribute names only. The default value for this attribute is `false`.

For the entry to match the filter, the presence of objectclass filter is used as follows:

```
<filter>
   <present name="objectClass"/>
</filter>
```

and is followed by the list of desired attributes:

```
<attributes>
   <attribute name="namingContexts"/>
   <attribute name="supportedLDAPversion"/>
   <attribute name="vendorName"/>
   <attribute name="vendorVersion"/>
   <attribute name="supportedSASLMechanisms"/>
</attributes>
```

Index

A

access
  anonymous 169, 170
  determining general types of 169
  precedence rule 195
access control
  ACI attribute 190
  password protection and 179
  roles 217
access control information (ACI) 189
  bind rules 190, 192, 193
  filtered rules 197
  format 190–194
  permission 190
  target 190, 191
  where to place 197
access rights
  granting 166
account inactivation 174
account lockout 187
ACI attribute 190
ACI instruction
  password protection and 179
ACI. See access control information
Administration Server
  master agents and 231
agents
  subagent 231
anonymous access 169, 170
  for read 35
  overview 170
applications 28
attribute
  ACI 190
  defining in schema 47
  required and allowed 53
  values 54
attribute-data pair 21
attributes
  naming 45
audits, for security 167
authentication methods 169
  anonymous access 170
  proxy authorization 172
  simple password 171

B

backup
  binary 265
  methods 265
  planning 265
  to ldif 268
bak2db 270
binary backup 265
binary restore 270
bind rules 190, 192, 193
branch point
  DN attributes 63
  for international trees 89
  for replication and referrals 64

network names 64

C

c attribute 89
cascading replication 134
chained suffixes 108
chaining 108–109
  and referrals 110
  roles limitation 76
change log 120
checking password syntax 178
class of service (CoS)
  access control 89
  cache 89
  classic 87
  filtered role limitation 89
  indirect 85
  limitations 88
  pointer 84
  template entry 82
classic CoS 87
clients
  bind algorithm 171
cn attribute. See commonName attribute
commonName attribute 53, 68, 70
consumer replica 117
consumer server 119
  role 119
CoS template entry 82
country attribute 89, 197
custom schema files 49

D

data
  accessing 24, 35
  backing up 265, 268
  consistency 52
  examples of 22
  management 149
  ownership 34
  planning 21
  privacy 167
  restoring 270
data master 32
  across multiple applications 33
  for replication 32
database
  chaining 94
  LDBM 94
  multiple 95
db2bak 265
db2ldif 268
default referrals 101
default schema
  customizing 43
  extending 44
  mapping data to 41
  viewing 41
deleting schema 48
design process 18
directory applications 28
  browsers 28
directory data
  accessing 24, 35
  examples of 22
  mastering 32
  ownership 34
  planning 21
directory design
  overview 17–19
directory server
  piloting 19
directory tree
  access control considerations 66
  branch point
    DN attributes 63
    for international trees 89
    for replication and referrals 64
    network names 64
  branching 61
  creating structure 60
  design
    choosing a suffix 58

examples
international enterprise 89
ISP 90
replication considerations 64
disaster recovery 142
distinguished name
collisions 68
DIT. See directory tree
DN name collisions 68
DSML 24
DSRK tools, downloading 12
dynamic groups 72

E

encryption
password 179
Salted SHA 179
SHA 179
entries
naming 67
non-person 70
organization 69
person 68
entry distribution 94
multiple databases 94
suffixes 96
expiration of passwords
overview 177
warning message 178
extending the schema 44

F

failure 264
filtered access control rules 197

G

group attribute 197
groups
dynamic 72
static 71

H

high availability 146, 147
hub replica 117
hub supplier 134

I

indirect CoS 85
inetOrgPerson attribute 197
installation location 11

L

LDAP referrals 101
load balancing 148

M

mail attribute 68
managed devices 231
managed object 232
master agent 231
master replica 117
multi-master replication 127–134
multiple databases 95

N

naming entries 67
  organization 69
  people 68
network management station (NMS) 232
network names, branching to reflect 64
network, load balancing 148

O

object classes
  defining in schema 46
  naming 45
  standard 40
object identifiers. See OIDs
OID registry 45
OIDs
  obtaining and assigning 45
organization attribute 197
organizationalPerson object class 53
organizationalUnit attribute 197

P

password policies 179
  change after reset 177
  design 175
  expiration warning 178
  password expiration 177
  password history 179
  password length 178
  password storage scheme 179
  replication of 187
  syntax checking 178
password storage scheme
  configuring 179
passwords
  changing after reset 177
  encryption of 179
  expiration 177
  expiration warning 178
  history 179
  minimum length 178
  reusing 179
  simple 171
  syntax checking 178
PDUs 232
performance
  replication and 135
permissions
  allowing 195
  bind rules 190, 192, 193
  denying 195
  on ACIs 190
  precedence rule 195
person entries 68
pointer CoS 84
precedence rule 195
protocol data units. See PDUs
proxy authentication 172
proxy authorization 172
proxy DN 172

R

referrals 100–108
  and chaining 110
  branching to support 64
  default 101
  LDAP 101
  smart referrals 103
replicas 117
  consumer 117
  hub 117
  master 117
replication 115–158
  access control 155
  branching to support 64
  cascading 134
  change log 120
  consumer server 119
  consumer-initiated 118
  data consistency 124

data master 32
database links 157
disaster recovery 142
high availability 147
hub server 134
load balancing 148
local availability 147
local data management and 149
maintaining schema consistency in 54
overview 115
password policies 187
performance 135
replication manager 120
resource requirements 145
schema 157
server plug-ins 155
single-master 125
site survey 144
strategy 142
supplier bind DN 120
supplier-initiated 118
replication examples
large sites 154
load balancing 152
small sites 153
replication manager 120
replication topologies 273–292
five data centers 286
one data center 273
three data centers 282
two data centers 279
using retro changelog 290
restore
binary 270
restoring data 270
reusing passwords 179
roles 72–77
access control 217
chaining limitation 76
compared to groups 77
CoS limitation 77
limitations 76
root suffix 96

S

Salted SHA encryption 179
schema
adding new attributes 47
assigning OIDs 45
checking 52
consistency 52–54
custom files 49
customizing 43
deleting elements 48
designing 41
extending 44
mapping date to 41
matching data to 42
naming attributes 45
naming elements 45
naming object classes 45
object class strategies 46
standard 40–??
viewing default 41
schema replication 157
security
conducting audits 167
security methods 168
security policy 36
security threats 164
denial of service 165
unauthorized access 164
unauthorized tampering 165
serverRoot 10
SHA encryption 179
Simple Network Management Protocol. See SNMP
simple password 171
single-master replication 125
site survey 27
availability requirements 32
characterizing data 31
documenting 37
identifying access methods 30
identifying applications 28
identifying data sources 30
network capabilities 144
smart referrals 103
sn attribute 53

SNMP
agents 231
managed devices 231
managed objects 232
master agent 231
NMS-initiated communication 232
overview 231
subagent 231
standard object classes 40
standard schema 40
static groups 71
streetAddress attribute 53
sub suffix 96
subagents 231
suffix
naming conventions 59
root suffix 96
sub suffix 96
supplier bind DN 120
surname attribute 53
syntax
password 178

T

telephoneNumber attribute 53
template entry. See CoS template entry.
topology
overview 93

U

uid attribute 53, 68
user authentication 171
userPassword attribute 53

W

warning, password expiration 178