Sun Java™ System

Directory Server 5.2 Technical Overview

2005Q1

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

Part No: 817-7619-10

Contents

List of Figures **7**

Preface **9**

Conventions 9
Related Books 12
Directory Server Books 12
Administration Server Books 12
Directory Proxy Server Books 12
Related Java Enterprise System Books 12
Documentation, Support, and Training 13
Related Third-Party Web Site References 13
Sun Welcomes Your Comments 13

Chapter 1 What's New in Directory Server 5.2 2005Q1 . . . . . **15**
Renaming and Moving Entries 15
Retro Change Log 15
Using the Retro Change Log With Replication 15
Logging Updates to Specific Suffixes on a Server 16
Logging Specified Attributes of a Deleted Entry 17

Chapter 2 Introduction to Directory Services and Directory Server **19**
What Is a Directory Service? 19
About Directory Services 20
About Enterprise-Wide Directory Services 21
About LDAP 22
About DSML 22
Tuned for Enterprise and e-business Directory Roles 23

What Is Directory Server? 24
Directory Server and Java Enterprise System 24
Directory Server Architecture Overview 25
LDAP and DSML Front Ends 26
Support for Industry Communication Standards 26
Directory Server Plug-in Extensibility 27
Sun Java System Directory Server Resource Kit 28
Java Naming and Directory Interface™ (JNDI) 29
Directory Information Tree 29
Multiple Database Design and Large Cache Support 30
Sun Cluster Agent 3.1 Support 31
Directory Server Data Storage 31
About Directory Entries 31
Distributing Directory Data 32
Directory Server Data Management 32
Importing Data 32
Exporting Data 33
Backing Up and Restoring Data 33
Indexing Data 33
Directory Server Schema 34
Schema Format 34
Standard Attributes 35
Standard Object Classes 36

Chapter 3 Directory Server Performance 37
Using Indexes for Efficient Searches 37
Performance Enhancements 38
Enhanced Update Performance 38
Enhanced Search Performance 39
Enhanced Replication Performance 39
Enhanced SSL Performance with Crypto Accelerator 1000 Board 40

Chapter 4 Directory Server Availability 41
Replicating Your Directory Server 41
Replication Concepts 41
Possible Replication Configurations 45
Business Scenarios and Their Associated Replication Solutions 48
Backup and Restoration Possibilities 51
High Availability Support 53

Chapter 5 Directory Server Scalability 55
Multiple Database, Multiple Server, and Data Distribution Possibilities 55

Distributing Data Across Multiple Databases and Servers 55
How to Manage Your Distributed Data 56
Scalable Data Management 57
Scalable Grouping of Entries 57
Scalable Attribute Management 60
Effective Rights Management 61
Scalable Schema Management 61
Directory Proxy Server and Scalability 62

Chapter 6 Directory Server Security 65
Directory Server Security Overview 65
Authentication and Account Inactivation 66
Controlling Access 67
Access Control - The Basics 68
Macro ACIs 69
Requesting Effective Rights Information 69
ACIs - The Tips 69
Password Policy 69
Password Policy Overview 69
Configuring Password Policies 71
Preventing Dictionary-Style Attacks 72
Controls for Resetting Passwords 72
Securing Connections and Data Storage 72
SSL Encryption and Authentication 73
Start Transport Layer Security 73
SASL Encryption and Authentication 74
Attribute Encryption 74
Denial of Service Attacks 75

Chapter 7 Directory Server Manageability 77
Server Management Console 77
Extensive Monitoring Possibilities 79

Glossary 81

Index 83

List of Figures

Figure 2-1 Sample Directory Tree 30
Figure 4-1 Cascading Replication Scenario 46
Figure 4-2 Multi-Master and Cascading Replication Scenario 47
Figure 7-1 Replication Nodes in Directory Server Console With Their Replication Enabled and Replication Disabled Icons 78
Figure 7-2 Replication Panel on the Directory Server Console Configuration Tab 78

Preface

This overview serves as a one-stop port of call for users, namely IT managers, system administrators, and developers needing to gain a rapid understanding of how Sun Java™ System Directory Server can address their business and programming needs. The guide walks users through Directory Server architectural concepts and outlines the benefits to be gained from Directory Server functionality in terms of the performance, reliability, and scalability provided in key Directory Server areas.

For information about how to access Sun™ documentation and how to use Sun documentation, see the following sections:

- Conventions
- Related Books
- Documentation, Support, and Training
- Related Third-Party Web Site References
- Sun Welcomes Your Comments

Conventions

Table 1 describes the typeface conventions used in this document.

Table 1 Typeface Conventions

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

Table 1 Typeface Conventions *(Continued)*

Typeface	Meaning	Examples
AaBbCc123 (Monospace bold)	What you type, as contrasted with on-screen computer output.	% **su** Password:
AaBbCc123 (Italic)	Book titles.	Read Chapter 6 in the *Developer's Guide*.
	New words or terms.	These are called *class* options.
	Words to be emphasized.	You *must* be superuser to do this.
	Command-line variables to be replaced by real names or values.	The file is located in the *ServerRoot* directory.

Table 2 describes placeholder conventions used in this guide.

Table 2 Placeholder Conventions

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

Table 3 describes the symbol conventions used in this book.

Table 3 Symbol Conventions

Symbol	Meaning	Notation	Example
[ ]	Contain optional command options.	O[*n*]	O4, O

Table 3 Symbol Conventions *(Continued)*

Symbol	Meaning	Notation	Example
{ }	Contain a set of choices for a required command option.	d{y|n}	dy
|	Separates command option choices.		
+	Joins simultaneous keystrokes in keyboard shortcuts that are used in a graphical user interface.		Ctrl+A
-	Joins consecutive keystrokes in keyboard shortcuts that are used in a graphical user interface.		Esc-S
>	Indicates menu selection in a graphical user interface.		File > New File > New > Templates

Table 4 describes the shell prompt conventions used in this book.

Table 4 Shell Prompts

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

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

Related Books

The following books can be found in HTML and PDF at http://www.sun.com/documentation/.

Directory Server Books

Directory Server Release Notes

Directory Server Technical Overview

Directory Server Deployment Planning Guide

Directory Server Installation and Migration Guide

Directory Server Performance Tuning Guide

Directory Server Administration Guide

Directory Server Administration Reference

Directory Server Plug-in Developer's Guide

Directory Server Plug-in Developer's Reference

Directory Server Man Page Reference

Administration Server Books

Administration Server Release Notes

Administration Server Administration Guide

Administration Server Man Page Reference

Directory Proxy Server Books

Directory Proxy Server Release Notes

Directory Proxy Server Administration Guide

Related Java Enterprise System Books

Java Enterprise System Installation Guide

Java Enterprise System Upgrade and Migration Guide

Java Enterprise System Glossary

Documentation, Support, and Training

Table 5 provides links to Sun documentation, support, and training information.

Table 5 Documentation, Support, and Training links

Typeface	Meaning	Examples
Documentation	`http://www.sun.com/documentation/`	Download PDF and HTML documents, and order printed documents.
Support and Training	`http://www.sun.com/supportraining/`	Obtain technical support, download patches, and learn about Sun courses.

Related Third-Party Web Site References

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

Sun Welcomes Your Comments

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

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

Please provide the full document title and part number in the appropriate fields. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the part number of this document is 817-7619-10.

Chapter 1

What's New in Directory Server 5.2 2005Q1

This chapter describe the new features of Directory Server 5.2 2005Q1.

Renaming and Moving Entries

In versions of Directory Server prior to Directory Server 5.2 2005Q1, it was possible to rename an entry. In Directory Server 5.2 2005Q1 and onwards, it is possible to rename an entry or move an entry.

The modify DN operation cannot be used for the following tasks:

- Move an entry from one suffix to another suffix
- Rename or move the root suffix

For information about how to use the modify DN operation, see "Renaming and Moving Entries" in the *Directory Server Administration Guide*.

Retro Change Log

Using the Retro Change Log With Replication

In versions of Directory Server prior to Directory Server 5.2 2005Q1, the retro change log could not identify the order in which changes where made to each replica in a multi-master topology. Therefore, the retro change log could not be used in a multi-master replication environment.

In Directory Server 5.2 2005Q1, the retro change log identifies the order in which updates are made for each replica. The retro change log can now be used in a multi-master replication environment. However, the failover of the retro change log is possible under certain conditions only.

The following new attributes have been defined for this feature:

- changeIsReplFixupOp
- deletedEntryAttrs
- replicationCSN
- replicaIdentifier
- targetUniqueId

For information about these attributes, see the object class reference in the *Directory Server Administration Reference*.

For further information about replication and the retro change log, see the *Directory Server Deployment Planning Guide*.

Logging Updates to Specific Suffixes on a Server

In versions of Directory Server prior to Directory Server 5.2 2005Q1, when the retro change log was enabled on a server, it logged updates to all suffixes on the server.

In Directory Server 5.2 2005Q1, the retro change log can be configured to log updates to specified suffixes only. A configurable argument `nsslapd-pluginarg`*<0-9>* has been defined for this feature.

For information about this feature, see the following references:

- For information about how to use the feature, see “Configuring Retro Change Log to Record Updates to Specified Suffixes” in the *Directory Server Administration Guide*.
- For information about the `nsslapd-pluginarg2` argument, see “Server Plug-In Functionality Reference” in the *Directory Server Administration Reference*.

Logging Specified Attributes of a Deleted Entry

In versions of Directory Server prior to Directory Server 5.2 2005Q1, the retro change log did not log attributes of an entry when that entry was deleted.

In Directory Server 5.2 2005Q1, the retro change log can be configured to log specified attributes of an entry when that entry is deleted. The standard attribute `deletedEntryAttributes`, and a plug-in argument `nsslapd-pluginarg`*<0-9>* has been defined for this feature.

For information about this feature, see the following references:

- For information about how to use the feature, see "Configuring Retro Change Log to Record Attributes of a Deleted Entry" in the *Directory Server Administration Guide*.
- For information about the `deletedEntryAttributes` attribute, see "Attribute Reference" in the *Directory Server Administration Reference*.
- For information about `nsslapd-pluginarg` arguments, see "Server Plug-In Functionality Reference" in the *Directory Server Administration Reference*.

Chapter 2

Introduction to Directory Services and Directory Server

Directory Server provides a central repository for storing and managing information. Almost any kind of information can be stored, from identity profiles and access privileges to information about application and network resources, printers, network devices and manufactured parts. Information stored in Directory Server can be used for the authentication and authorization of users to enable secure access to enterprise and Internet services and applications. Directory Server is extensible, can be integrated with existing systems, and enables the consolidation of employee, customer, supplier, and partner information.

Directory Server provides the foundation for the new generation of e-business applications and Web services, with a centralized and distributed data repository that can be used in your intranet or over your extranet with your trading partners.

This chapter describes the basic concepts you must understand before attempting to design and deploy Directory Server. This chapter is divided into two sections:

- What Is a Directory Service?
- What Is Directory Server?

What Is a Directory Service?

This section defines Directory Services and enterprise-wide directory services, in addition to clarifying the differences between directories and databases. It examines the major roles played by directories and establishes exactly where Sun Java System Directory Server fits into the Directory Service picture. This section is divided into the following parts:

- About Directory Services

- About Enterprise-Wide Directory Services
- About LDAP
- About DSML
- Tuned for Enterprise and e-business Directory Roles

About Directory Services

Directory services are an essential part of today's network-centric computing infrastructure. Directory-enabled applications now power almost all the mission critical processes of an enterprise, including resource planning, value chain management, security and firewalls, and resource provisioning. Directory services also provide the foundation for deployment of e-business and extranet applications. So what exactly is a Directory Service?

A directory service is the collection of software and processes that store information about your enterprise, subscribers, or both. An example of a directory service is the Domain Name System (DNS), which is provided by DNS servers. A DNS server stores the mappings of computer host names and other forms of domain name to IP addresses. A DNS client sends questions to a DNS server about these mappings (e.g. what is the IP address of test.example.com?). Thus, all of the computing resources (hosts) become clients of the DNS server. The mapping of host names enables users of the computing resources to locate computers on a network, using host names rather than complex numerical IP addresses.

Whereas the DNS server stores only two types of information: names and IP addresses, an LDAP directory service can store information on many other kinds of real-world and conceptual objects. Sun Java System Directory Server stores all of these types of information in a single, network-accessible repository. You may for example want to store physical device information, employee information (name, E-mail address), contract or account information (name, delivery dates, contract numbers, etc.), authentication information, manufactured production information. It is worth noting that although a directory service can be considered an extension of a database, directory services generally have the following characteristics:

- *Hierarchical naming model*
  A hierarchical naming model uses the concept of containment to reduce ambiguity between names and simplify administration. The name for most objects in the directory is relative to the name of some other object which conceptually contains it. For example, the name of an object representing an employee of a particular company contains the name of the object representing the company, and the name of the company might contain the name of the

objects representing the country where the company operates, e.g. `cn=John Smith, o=Example Corporation, c=US`. Together the names of all objects in the directory service form a tree, and each Directory Server holds a branch of that tree, which in the Sun Java System Directory Server documentation is also referred to as a *suffix*.

- *Extended search capability*
  Directory services provide robust search capabilities, allowing searches on individual attributes of entries.
- *Distributed information model*
  A directory service enables directory data to be distributed across multiple servers within a network.
- *Shared network access*
  While databases are defined in terms of APIs, directories are defined in terms of protocols. Directory access implies network access by definition. Directories are designed specifically for *shared* access among applications. This is achieved through the object-oriented schema model. By contrast, most databases are designed for use only by particular applications and do not encourage data sharing.
- *Replicated data*
  Directories support replication (copies of directory data on more than one server) which make information systems more accessible and more resistant to failure.
- *Datastore optimized for reads*
  The storage mechanism in a directory service is generally designed to support a high ratio of reads to writes.
- *Extensible schema*
  The schema describes the type of data stored in the directory. Directory services generally support the extension of schema, meaning that new data types can be added to the directory.

About Enterprise-Wide Directory Services

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

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

An enterprise-wide directory service solves the n+ 1 directory problem by providing a single, centralized repository of directory information that any application can access. However, giving a wide variety of applications access to the directory requires a network-based means of communicating between the applications and the directory. Directory Server provides two ways in which applications can access its enterprise-wide directory:

- Lightweight Directory Access Protocol (LDAP)
- Directory Services Markup Language (DSML)

About LDAP

LDAP provides a common language that client applications and servers use to communicate with one another. LDAP-based applications can easily search, add, delete and modify directory entries. LDAP is a “lightweight” version of the Directory Access Protocol (DAP) defined in the ISO / ITU-T X.500 standard. DAP gives any application access to the directory via an extensible and robust information framework, but at an expensive administrative cost. DAP does not use the Internet standard TCP / IP protocol and has complicated directory-naming conventions. LDAP preserves the best features of DAP while reducing administrative burdens. LDAP uses an open directory access protocol running over TCP / IP and uses simplified encoding methods. It retains the X 500 standard data model and can support millions of entries for a comparatively modest investment in hardware and network infrastructure.

About DSML

DSML is a markup language that enables you to represent directory entries and commands in XML. This means that XML-based applications using HTTP can take advantage of directory services while staying within the existing web infrastructure. Directory Server implements version 2 of the DSML standard (DSMLv2). For detail regarding the restrictions and extensions to the DSML standard refer to the *Directory Server Administration Reference*.

Tuned for Enterprise and e-business Directory Roles

Online directories that support LDAP have become critical components of e-business infrastructure, supporting identity and risk management in several important roles. They provide a dynamic and flexible means of storing information and retrieving it over the internet, and can of course be configured to use the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols for authenticated and encrypted communications. As protected repositories of personal information, LDAP directories are also a key component for the personalized delivery of services to users of the directory and personalized treatment of information contained in the directory. Directories have the following three primary roles to play, each of which have very different deployment characteristics:

- *Network Operating System (NOS) Directory*

  A NOS is a distributed operating system that manages multiple requests concurrently in a multiuser environment. It consists of clients and servers configured so that client machines can access the drivers on servers as if they were local drives on the client machine, and servers can handle requests from the client to share files and applications as well as network devices such as printers or faxes.

  A NOS directory is used to administer a NOS allowing intranet users to log in once for all network file and printing requirements. The user population for a NOS directory ranges from 20 to 20,000 users, The NOS directory provides a single point of integrated administration and management for the network and the most important characteristics of this directory role are the tight integration with the NOS and sophisticated management tools for clients and servers.

- *Enterprise Directory*

  An enterprise directory provides a global repository for shared information about people, organizations, and devices, including white/yellow pages, organization charts, seating charts, information about network devices such as routers, etc. It is important for an enterprise directory to provide flexible ways of organizing data and flexible management tools that can accommodate heterogeneous application environments or autonomous business units (through delegated administration). The user population for an enterprise directory tends to range from between 20,000 and 200,000 users.

- *e-business Directory*

  An e-business directory maintains information about customers, trading partners, and suppliers, which involves making the information available to an extranet as necessary and appropriate. The user population for an e-business directory is typically millions of users, and in addition to the flexible management tools required the NOS and enterprise directories, e-business directories require a high degree of reliability and scalability to be able to rapidly accommodate millions of users.

The deployment characteristics of each role vary hugely, and as we will see throughout this Technical Overview, Sun Java System Directory Server is very much tuned for the enterprise and e-business directory roles, in that it provides a single directory service infrastructure for both enterprise and e-business applications and services. Sun Java System Directory Server is a mature LDAP directory solution which supports open standards, and it's carrier-grade design provides the extreme scalability, performance, and high availability required by large enterprise and e-business directories that support millions of users.

What Is Directory Server?

This section provides an overview of Sun Java System Directory Server by way of an introduction to the following chapters which outline the benefits to be gained from Directory Server in terms of performance, availability, scalability, security, and manageability. This section is divided into the following parts:

- Directory Server and Java Enterprise System
- Directory Server Architecture Overview
- Directory Server Data Storage
- Directory Server Data Management
- Directory Server Schema

Directory Server and Java Enterprise System

Sun Java System Directory Server provides a central repository for storing and managing intranet and Internet information such as identity profiles (employees, customers, suppliers, etc.), user credentials (public key certificates, passwords, and pin numbers), access privileges, application resource information, and network resource information. Directory Server is now delivered as a component product of the Sun Java Enterprise System, Sun's

software infrastructure that provides the services needed to support enterprise-strength applications distributed across a network or Internet environment. For further information on Sun Java Enterprise System, and on the concepts underlying distributed enterprise applications and distributed service infrastructures see the *Directory Server Technical Overview*..

Directory Server Architecture Overview

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

- An LDAP server for processing requests.
- Sun Java System Administration Server.
- Sun Java System Server Console for managing Directory Server.
- Command-line tools for starting and stopping the server, importing and exporting data in the database, database re-indexing, account inactivation and deactivation, LDIF merges, kernel tuning, and replication management.
- Front ends responsible for LDAP and DSMLv2.
- Agent responsible for responding to Simple Network Management Protocol (SNMP) queries about the state of the server.
- Plug-ins for server functions, such as access control and replication.
- An initial directory information tree, consisting of server configuration and sample enterprise data.

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

You can use Directory Server to manage extranet user-authentication, create access control, set up user preferences, and centralize user management. In hosted environments, partners, customers, and suppliers can manage their own areas of the directory, reducing administrative costs. We will examine the following items in more detail and expand on some of the more advanced architecture features of Directory Server:

- LDAP and DSML Front Ends
- Support for Industry Communication Standards

- Directory Server Plug-in Extensibility
- Sun Java System Directory Server Resource Kit
- Java Naming and Directory Interface™ (JNDI)
- Directory Information Tree
- Multiple Database Design and Large Cache Support
- Sun Cluster Agent 3.1 Support

LDAP and DSML Front Ends

The server front ends of Sun Java System Directory Server manage communications with directory client programs. Directory Server functions as a daemon and multiple client programs can communicate with the server using LDAP over TCP / IP or DSMLv2 over HTTP / SOAP. These connections can be made secure using the Secure Socket Layer over Transport Layer Security (SSL / TLS), depending on whether the client negotiates the use of TLS for the connection.

Multiple clients can bind to the server at the same time over the same network because Directory Server is a multi-threaded application. As your directory services grow to include larger numbers of entries or larger numbers of clients spread out geographically, they also include multiple Directory Servers placed in strategic locations around the network or organization. You can also distribute your Directory Servers to conform with data protection regulations and to provide high availability.

Another advantage of Directory Server is that it implements LDAP natively, thereby avoiding the performance and management overheads associated with having a gateway on top of an X.500 directory, and with relational databases.

Support for Industry Communication Standards

Directory Server supports two standard communication protocols: LDAP version 3 and DSML version 2.

LDAP Version 3

Directory Server supports LDAP versions 3 (RFC 2251), the definitive Internet Proposed Standard protocol for accessing directory information. LDAP was invented by the Internet research community and provides a common language that client applications and servers use to communicate with one another. LDAP-based applications can easily authenticate, search, add, delete, and modify directory entries.

Directory Server supports LDAP search filters as outlined in RFC 2254, including the following: presence, equality, inequality, substring, approximate match (for phonetic matching), greater than, less than, Boolean combinations of the previous filters using the and (&), or (|), and not (!) operators.

Directory Server also supports LDAP version 3 search references (also known as smart referrals), which allow the directory to refer a query to another directory. It also implements LDAP URL formats as outlined in RFC 2255. Directory Server uses LDAP Data Interchange Format (LDIF) for exchanging directory information (RFC 2849).

DSML Version 2

As we have seen in the About DSML section, DSML is a markup language that enables you to represent directory entries and commands in XML. Directory Server implements version 2 of the DSML standard (DSMLv2). DSML enables developers to combine the power of XML in presenting and manipulating data with the scalability, security, availability, and information management strengths of Directory Server.

Because DSML is not an access protocol, Directory Server uses HTTP and the SOAP version 1.1 to transport the DSML content. Directory Server supports DSML natively, providing very high throughput performance as opposed to the gateway design used with other LDAP directories.

Using DSML over HTTP/SOAP, you can create applications that do not rely on LDAP, allowing non-LDAP clients to interact with your directory data. This allows you to create and implement a new generation of Web services that interface with Directory Server through XML.

The DSML front end of Directory Server is not a full web or application server. In this version of Directory Server it only accepts SOAP operations for requests conforming to the DSML specification. Because the DSML front end is a core component of the directory, all native access controls apply. As a consequence, any all security rules that you previously defined for LDAP also apply to DSML, including simple authentication, SSL authentication, and Access Control Instructions (ACIs).

Directory Server Plug-in Extensibility

Directory Server relies on plug-ins for many key features. A plug-in is a way to add, enable, and configure functionality to the core server. For example plug-ins are included to support referential integrity and uniqueness constraints.

Plug-ins extend Directory Server functionality by adding pre or post operational functionality. You can add new intelligence to operations, giving you the ability to adapt your directory to other applications, changing business requirements, partners' needs, and so on.

The plug-in API is fully documented in this release of Directory Server, enabling you to create your own custom plug-ins. Refer to the *Directory Server Plug-in Developer's Guide* for help on how to develop plug-ins, upgrade existing plug-ins and explore sample plug-ins and the *Directory Server Plug-in Developer's Reference* for details about particular data structures, functions, parameter, and block semantics. Sun Professional Services and other organizations can help you to develop custom plug-ins. For information about what Sun Professional Services has to offer, refer to:

`http://www.sun.com/service/sunps/sunone`

Sun Java System Directory Server Resource Kit

The Sun Java System Directory Server Resource Kit (DSRK) provides tools and APIs for deploying, accessing, tuning, and maintaining your Directory Server. These utilities will help you implement and maintain more robust solutions based on LDAP, the Lightweight Directory Access Protocol.

The first two components of the DSRK are the LDAP SDKs (Software Development Kits) for C and Java™ programming languages. These SDKs make it simple to write client applications for your directory, and these APIs expose all of the functions for connecting to an LDAP directory and accessing or modifying its entries. Use them to design and integrate directory functionality into your applications at the programmatic level.

The third component of the Sun Java System Directory Server Resource Kit is the set of tools and scripts that make a directory accessible through a command-line shell. The wide range of tools can be used for simple directory access, performance testing, and for the maintenance of Directory Servers. These tools are themselves based on the LDAP SDKs, and they were created to help development teams to test and validate Directory Server. It is also worth noting that the commands that run these tools can be used to write scripts to automate all of these tasks.

For more information refer to the *Directory Server Resource Kit Tools Reference*.

Java Naming and Directory Interface™ (JNDI)

The Java Naming and Directory Interface (JNDI) is a standard extension to the Java platform, providing applications based on Java technology with a unified interface to multiple naming and directory services. You can build powerful and portable directory-enabled applications using this industry standard. This technology supports accessing Directory Server using LDAP and DSML v2 from Java applications. General information about JNDI is available from:

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

A JNDI Tutorial containing detailed descriptions and examples of how to use JNDI is available at:

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

Directory Information Tree

Directory Server contains a basic directory information tree at installation time. This tree mirrors the tree model used by most file systems, with the tree's root, or first entry, appearing at the top of the hierarchy. The root of this tree is called the root suffix. At installation time the directory contains three subtrees under the root suffix:

- `cn=config`

where `cn` stands for Common Name. This subtree contains information about the server's internal configuration.

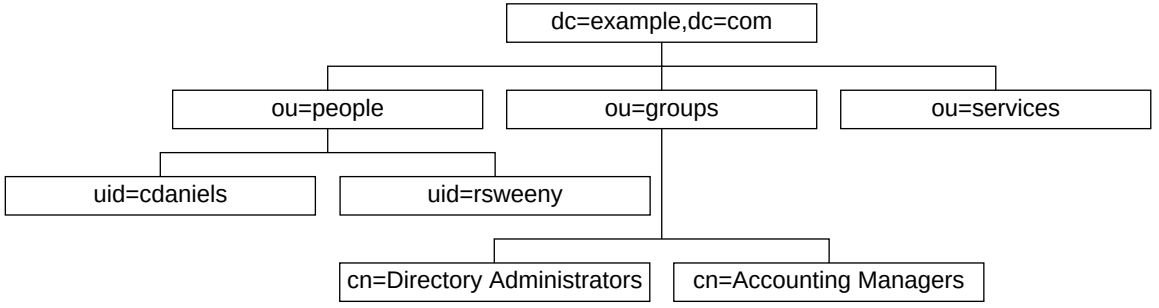
- `o=NetscapeRoot`

where `o` stands for organization. This subtree contains the configuration information of other Sun Java System servers, such as Sun Java System Administration Server which takes care of authentication and all actions that cannot be performed through LDAP (such as starting or stopping Directory Server). This subtree name originates from a legacy version of the product.

- `o=UserRoot`

During installation, a user database is created by default which is called `UserRoot`. An example name for the suffix stored on this user database might be `o=UserRoot`. You can choose to populate this database at installation time, or at a later stage. You can build on the default directory tree to add any data relevant to your directory installation. In Figure 2-1, the `o=UserRoot` suffix has been renamed to `dc=example,dc=com`, and additional subtrees have been added to reflect the organizational hierarchy.

Figure 2-1 Sample Directory Tree



In the preceding figure:

- `dc` refers to the domain component
- `ou` refers to the organizational unit
- `uid` refers to the user id

For further detail about the directory information tree refer to the *Directory Server Deployment Planning Guide*.

Multiple Database Design and Large Cache Support

The multiple database architecture of Directory Server supports distributed naming contexts, providing large scalability to support millions of users on a single system, improved backup and restore, load balancing, and simplified administration. The multiple database design also provides a method for partitioning the directory data to simplify replication.

Directory Server can run as a 64-bit application on Solaris SPARC®. This Directory Server, using the large cache allows you to attain improved performance for high-volume deployments.

Sun Cluster Agent 3.1 Support

The Sun Cluster Agent is a high availability agent that provides LDAP service failover. It is bundled with Directory Server and on certain hardware and deployment configurations, can be used to improve the availability of Sun Java System services provided by multiple servers. For more information on how to install the Agent see the *Directory Server Installation and Migration Guide*.

Directory Server Data Storage

By default, Directory Server uses a single database to store the directory tree. This database can manage millions of entries. The default database supports advanced methods of backing up and restoring data.

Since Directory Server supports multiple databases you can also distribute data across the databases, enabling the server to store more data than can be held in a single database.

The following sections describe how a directory database stores data.

About Directory Entries

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

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

Directory Server also uses read-only attributes that are either calculated by the server, or for certain server functions such as access control, set by the administrator. These attributes are called *operational attributes*.

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

Note that not all entries are automatically returned in response to an LDAP search. For instance, entries of the `ldapsubentry` object class are not returned in response to normal search requests. An `ldapsubentry` entry represents an administrative object, for example the entries that are used internally by Directory Server to define a role or a class of service. To receive these entries, clients must search specifically for entries of the `ldapsubentry` object class.

Distributing Directory Data

When you store various parts of a tree in separate databases, your directory can process client requests in parallel, improving performance. You can also store databases on different machines, to improve performance further. We examine in further detail the various data distribution options provided in Chapter 5, “Directory Server Scalability”.

Directory Server Data Management

The database is the basic unit of storage, performance, replication, and indexing. A variety of operations can be performed on a database, including importing, exporting, backing up, restoring, and indexing.

These command line utilities are subcommands of the `directoryserver` command. For more information see the *Directory Server Man Page Reference*.

Importing Data

Directory Server provides three methods for importing data:

- Importing from the Directory Server Console.

  You can use the Directory Server Console to append data to all of your databases, including database links.

- Initializing databases.

  You can use the Directory Server Console to import data to one database. This method overwrites any data contained by the database.

- Importing data from the command line.

You can import data using the command-line utilities `ldif2db`, `ldif2db-task`, and `ldif2ldap`.

Exporting Data

You can use LDIF to export database entries from your databases. LDIF is a standard format described in RFC 2849, "The LDAP Data Interchange Format (LDIF) - Technical Specification."

Exporting data can be useful for the following:

- Backing up the data in your database
- Copying your data to another Directory Server
- Exporting your data to another application
- Repopulating databases after a change to your directory topology

You can use the Directory Server Console or the command-line utilities `db2ldif` and `db2ldif-task` to export data.

Backing Up and Restoring Data

You can use Directory Server Console or the `db2bak` or `db2bak-task` command line utilities to back up directory data. Both methods allow you to perform a backup while the server is running, which prevents you having a period during which the directory is not accessible.

You can restore data from a previously generated backup using Directory Server Console or the command-line utilities `bak2db` or `bak2db-task`. Restoring databases overwrites any existing database files. While restoring databases, the server must be running. However, the databases are unavailable for processing operations during the restore.

Indexing Data

Depending on the size of your databases, searches performed by client applications can take a lot of time and resources. You can use indexes to improve search performance. Indexes are files stored in the directory databases. Separate index files are maintained for each database in the directory. Each file is named according to the attribute it indexes. The index file for a particular attribute can contain multiple types of indexes, allowing you to maintain several types of index for each attribute. For example, a file called `givenName.db3` contains all the indexes for the `givenName` attribute.

Depending on the types of applications using your directory, you will use different types of index. Different applications may frequently search for a particular attribute, or may search your directory in a different language, or may require data in a particular format. For more information on how to use indexes to improve your Directory Server performance see Chapter 3, “Directory Server Performance” on page 37.

Directory Server Schema

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

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

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

Schema Format

Directory Server bases its schema format on version 3 of the LDAP protocol (LDAPv3). This protocol requires Directory Servers to publish their schemas through LDAP itself, allowing directory client applications to 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`. In addition, it uses a private field in the schema entries called X-ORIGIN, which describes where the schema entry was defined originally. For example, if a schema entry is defined in the standard LDAPv3 schema, the X-ORIGIN field refers to RFC 2252. For example, the standard `person` object class appears in the schema as follows:

```
objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard LDAP
objectclass' SUP top MUST ( sn $ cn ) MAY ( description $ seeAlso
$ telephoneNumber $ userPassword ) X-ORIGIN 'RFC 2256' )
```

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`).

Standard Attributes

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

```
cn: Charlene Daniels
```

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

```
dn: uid=cdaniels, ou=people, dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Charlene Daniels
sn: Daniels
givenName: Charlene
givenName: Charlie
mail: cdaniels@example.com
```

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

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

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

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

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

The SYNTAX is the OID of the syntax for values of the attribute. For more information about the LDAPv3 schema format, refer to the LDAPv3 Attribute Syntax Definitions document (RFC 2252).

Standard Object Classes

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

Each directory entry belongs to one or more object classes. Once you place an object class identified in your schema on an entry, you are telling Directory Server that the entry can have a certain set of attribute values and must have another, usually smaller, set of attribute values.

Object class definitions contain the following information:

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

For further reading on schema see the *Directory Server Administration Reference*.

Chapter 3

Directory Server Performance

Since Directory Server provides the foundation for the new generation of e-business applications and Web services, it is essential that strong performance be maintained. This chapter examines Directory Server indexing which helps boost directory performance and other substantial performance improvements brought in this release. This chapter is divided into the following sections:

- Using Indexes for Efficient Searches
- Performance Enhancements

Using Indexes for Efficient Searches

Directory Server implements indexes so that you can search quickly the data in your directory. The index files are stored in the directory's databases. Directory Server supports the following types of indexes:

- Presence index: this index contains a list of the entries that contain a particular attribute. For example, you can use this type of index for examining any entries that contain access control information.
- Equality index: this index allows you to search efficiently for entries containing a specific attribute value.
- Approximate index: this index allows efficient approximate searches. For example, a search again locality~=San Fransisco (note the misspelling) would return entries including locality=San Francisco.
- Substring index: this index allows efficient searching against substrings within entries.

- International index: this index speeds up searches for information in international directories. During a search operation, you can request that the directory sort the results based on any language for which the server has a supporting collation order. The international index associates the object identifier (OID) of a locale with the attribute to be indexed.
- Browsing (virtual list view) index: this index speeds up the display of entries in the Sun Java System Server Console. The browsing index display entries in an order you configure, presenting them as small groups rather as a large, unorganized list. You can create a browsing index on any branch of your directory tree to improve the display performance.

You can use separate, specialized indexes for masters and consumers in a replication scenario. For example, a master replica might contain a UID index, a series of consumers that are used for phone number look-ups might contain a phone number index, another consumer might contain an E-mail address index while replication hubs might contain only system indexes (that is indexes that cannot be deleted or modified as they are required for Directory Server to function properly and efficiently).

Through the use of separate indexes, you can ensure that each machine has indexes that are optimized for its role in the overall directory architecture.

Performance Enhancements

Directory Server incorporates enhancements in terms of:

Enhanced Update Performance

Enhanced Search Performance

Enhanced Replication Performance

Enhanced SSL Performance with Crypto Accelerator 1000 Board

Enhanced Update Performance

The update performance enhancements include the following:

- Group flush: Directory Server can be configured to increase the speed of update operations by waiting for several operations to be completed before writing them to the physical disk and sending the acknowledgment back to the client application.

Usually, when an update is made, the modified data is written to the physical disk device. However, writing to disk has a higher latency than writing to memory. You can use hardware such as the Sun Storage T3 Array which include an extra level of cache to improve write performances.

- Index compression. This enhancement improves performance on large databases with large index lists and does not require specific configuration.
- Replication compression. On Solaris and Linux systems that communicate over a WAN, replication can be compressed to maximize the throughput on the WAN connection. This feature automatically optimizes bandwidth usage on the WAN and speeds up replication.
- Improved checkpointing. Checkpoints and updates are now carried out in parallel.

Enhanced Search Performance

The search performance improvements include:

- 64-bit server process. The entry, database, and import caches can now all be simultaneously larger than 4 GB. This means that the process size is no longer a concern.

  With the increased cache size, searches now scale almost linearly on servers with up to 12 processors.
- Improved algorithm for reading from the database cache to the entry cache. Performance is improved through reduced memory allocation and improved thread management.

Enhanced Replication Performance

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. Now that Directory Server now supports MMR over WAN, geographical boundaries no longer constitute a stumbling block for multi-master replication.

NOTE Due to differences in protocol, multi-master replication over WAN is not compatible with releases of Directory Server prior to Directory Server 5.2. Therefore, in a multi-master replication over WAN configuration, *all* Directory Server instances separated by a WAN *must* be of the following version: Sun ONE Directory Server 5.2, or Sun Java Enterprise System Directory Server 5.2 2003Q4 or later.

To optimize the replication flow and reduce round-trip delay time, Directory Server now:

- sends multiple replication operation requests rather than having to send them individually,
- specifies 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,
- and improves bandwidth performance by reducing the amount of data sent over the wire through smart compression and other internal enhancements, such as full asynchronous support.

Enhanced SSL Performance with Crypto Accelerator 1000 Board

Directory Server supports the Sun Crypto Accelerator 1000 Board (on Sun SPARC® hardware only). This board speeds up the initial exchange of keys (i.e. not the bulk encryption itself) by performing specific SSL mathematical functions, leaving the system processor to focus on application processing, which may be useful in deployments where client applications repeatedly bind over SSL, search, and then unbind. It is worth noting that the Sun Crypto Accelerator 1000 Board is most effective if the clients that are establishing connections are doing so from different machines. For further detail on the Sun Crypto Accelerator 1000 Board see the Securing Connections With SSL section in the *Directory Server Deployment Planning Guide*.

Chapter 4

Directory Server Availability

For your directory service to be successful it must of course be highly available. This chapter examines the replication mechanism offered by Directory Server whereby directory data is automatically copied from one directory to another, Directory Server's backup strategies and possibilities, as well as the high availability support provided by Sun Cluster 3.1 Agent. This chapter is divided into the following sections:

- Replicating Your Directory Server
- Backup and Restoration Possibilities
- High Availability Support

Replicating Your Directory Server

This section examines:

- Replication Concepts
- Possible Replication Configurations
- Business Scenarios and Their Associated Replication Solutions

and provides you with a solid understanding of what Directory Server has to offer in terms of replication.

Replication Concepts

Replication is the mechanism that automatically copies directory data from one Directory Server to another. Using replication you can copy any directory tree or subtree (stored in its own suffix) between servers. Replication enables you to provide a highly available directory service, and to distribute data geographically. In practical terms replication provides the following benefits:

Fault tolerance and failover

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

Reduced response time by load balancing

By replicating directory tree across servers, you can reduce the access load on a given machine, thereby improving server response time. Replication is *not* however a solution for write scalability (which must instead be achieved by data partitioning).

Reduced response time by localizing data

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

Local data management

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

This section sets out to provide you with a basic understanding of Directory Server replication concepts and examines following:

- Replica
- Suppliers and Consumers
- Replication Agreement
- Online Replica Promotion and Demotion
- Consumer Initialization and Incremental Updates
- Data Consistency

Replica

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

- Master, or read-write, replica: a read-write database that contains a master copy of the director data. A master replica can process update requests from directory clients.
- Consumer replica: a read-only database that contains a copy of the information held in the master replica. A consumer replica can process search requests from directory clients but refers update requests to master replicas.

- Hub replica: a read-only database, like a consumer replica, but stored on a Directory Server that supplies one or more consumer replicas.

You can configure Directory Server to manage several replicas, and each replica can play a different role in replication.

Suppliers and Consumers

A Directory Server that replicates to other servers is called a *supplier*. A Directory Server that is updated by other servers is called a *consumer*. The supplier replays all the updates on the consumer through specially designed LDAP v3 extended operations.

A server can be both a supplier and a consumer in the following cases:

- When the server contains a hub replica, that is receives updates from a supplier and replicates the changes to consumer(s). For more information refer to "Cascading Replication" on page 45.
- In multi-master replication, when a master replica is mastered on two or more different Directory Servers, each server acts as a supplier and a consumer of the other server. For more information refer to "Multi-Master Replication" on page 46.
- When the server manages a combination of master replicas and consumer replicas.

A server that plays the role of a consumer only, i.e. only contains a consumer replica, is called a *dedicated consumer*.

Replication Agreement

Directory Server uses *replication agreements* to define how replication occurs between two servers. It identifies amongst other things, the suffix to replicate, the consumer server to which the data is pushed, the times during which replication can occur, how the connection is secured (SSL or not). Note that a replication agreement describes replication between *one* supplier and *one* consumer. The replication agreement is configured on the supplier, and must be *enabled* for replication to work. It is possible to enable or disable existing replication agreements, which can be useful should you currently have no need for a replication agreement, but want to maintain its configuration for future use or backup possibilities.

Online Replica Promotion and Demotion

Directory Server enables you to promote and demote replicas online. Promoting or demoting a replica changes its role in the replication topology. Dedicated consumers may be promoted to hubs, and hubs may be promoted to masters. In the same way masters may be demoted to hubs, and hubs demoted to consumers. Note that both the promotion and demotion procedures are of necessity incremental: for promotion you must go from consumer to hub, then hub to master replica and vice versa for online demotion.

Online replica promotion and demotion provides increased flexibility and failover capabilities.

Consumer Initialization and Incremental Updates

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

The replication updates sent to the consumer as the modifications are made (that is, the updates that follow the initialization) are called incremental updates. Provided that the updates originate from different replicas, Directory Server allows a consumer to be incrementally updated by several suppliers at once.

Data Consistency

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

Directory Server provides the option of keeping replicas always synchronized, or of scheduling updates for a particular time of day, or day of the week. The advantage of keeping replicas always in sync is that data remains consistent across your topology. The cost, however, is the network traffic resulting from the frequent update operations. It is for you to decide what best suits your needs in terms of data consistency and your possibilities in terms of network connections and network traffic loads.

Possible Replication Configurations

Directory Server supports single master, cascading, and multi-master replication, all of which ensure the high availability of directory services for both read and write operations. Directory Server also provides a new replication feature, called fractional replication, for content security. We will examine the following configuration possibilities in this section:

- Single Master Replication
- Cascading Replication
- Multi-Master Replication
- Multi-Master Replication over Wide Area Networks (WAN)
- Fractional Replication

Single Master Replication

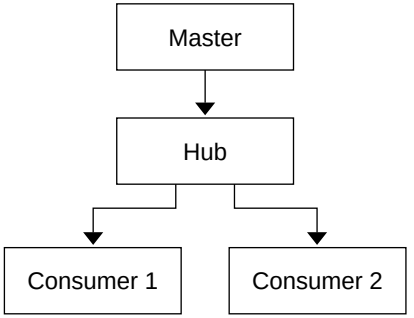
In the most basic replication configuration, a supplier copies a master replica directly to one or more consumers. In this configuration, all directory modifications are made to the master replica, and the consumers contain read-only copies of the data. All modifications are propagated to the consumer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

A single master replication configuration can be useful if you have a suffix that receives a large number of search and update requests from clients, as it allows you to distribute the search request load to the consumer which frees up the supplier for the update requests. Note that a supplier can replicate to several consumers, allowing you to distribute loads yet further, with the total number of consumers that a single supplier can manage depending on the speed of your network and the total number of entries that are modified on daily basis.

Cascading Replication

In cascading replication, Directory Server acts as a hub supplier. A hub is a read-only database, like a consumer replica, however, a hub also accepts replication from one or more master replicas and replicates the changes to consumer servers. Figure 4-1 illustrates a basic cascading replication scenario:

Figure 4-1 Cascading Replication Scenario



In a single master replication scenario, with one master and many consumers, the master's resources are consumed by replicating information. Using cascading replication, the master concentrates on handling operations, while the hub handles replicating data to all of the consumers.

You primarily use cascading replication to balance heavy traffic loads, reduce connection costs with local hubs in a geographically distributed environment, and increase performance. Using cascading replication, you can optimize the use of your hardware resources.

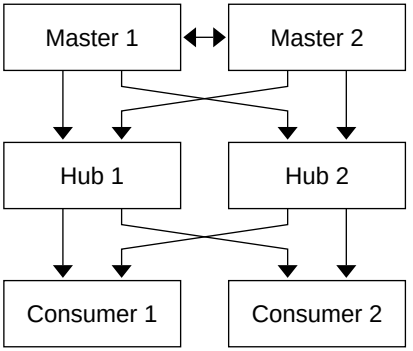
You can tune your cascaded architecture so that it optimizes the indexes, logs, and cache size of your Directory Server. For example, you can optimize the indexes on each server so that you have an update index for the supplier, a common name index for consumer 1 and a phone number index for Consumer 2.

Multi-Master Replication

Directory Server supports four-way multi-master replication over wide area networks. This means that in multi-master replication, a master replica is available on up to four Directory Servers. A master replica is a read-write database that contains a master copy of the directory data. Each master acts as a supplier and a consumer to the other Directory Servers.

When combined with single master and cascading replication scenarios, multi-master replication provides a highly flexible and scalable replication environment for enterprises and service providers with global data center operations. Figure 4-2 illustrates a multi-master and cascading replication scenario:

Figure 4-2 Multi-Master and Cascading Replication Scenario



Four-way multi-master replication is ideal for distributed deployments. For example, an enterprise could establish two masters in San Francisco and two masters in New York. If something happened to both of the masters at a single location, the two masters at the other location can continue to provide the service.

Multi-master deployments provide 24x7 service levels with write failover. For example, if one server fails, the others remain available for writes. When the server comes back online, it receives replication updates from the other masters.

The multi-master replication protocol is streamlined, allowing you to:

- Replicate updates based on the replica ID. The replica ID makes it possible for a consumer to receive updates for different replica IDs from multiple suppliers at the same time, improving performance.

  This method also ensures the replication of only the updates that come from a replica ID for which the supplier and consumer are not yet synchronized. Only one supplier can replicate changes for a replica ID at a given time for a particular consumer.

- Enable or disable a replication agreement with a particular consumer, giving you more flexibility in configuring how you deploy replication. For example, you can use this feature to configure a fully connected topology and use only part of it.

  You can enable and disable a replication agreement remotely for a particular consumer by modifying a single attribute value. By default, replication agreements are enabled.

You can implement a fully connected four-way multi-master replication topology, guaranteeing replication even if one or more masters fail. This type of deployment is appropriate for systems that have stringent availability requirements.

In terms of data consistency, multi-master replication uses a loose consistency replication model. This means that the same entries can be changed on different servers. When replication occurs between the two servers, the conflicting changes need to be resolved. Resolution occurs automatically, based on the timestamp associated with the change on each server.

Multi-Master Replication over Wide Area Networks (WAN)

The improved replication mechanism provided by Directory Server allows you to distribute directory databases across machines or network boundaries. The WAN, despite the challenges it introduces in terms of higher latency, lower bandwidth and a potentially higher number of errors (such as disconnections, packet loss, packets out of order, and congestion), no longer constitutes a stumbling block to replication configurations. For more detail on these replication performance enhancements see "Enhanced Replication Performance" on page 39.

Fractional Replication

In Directory Server, fractional replication allows you to replicate a subset of attributes for all the entries in a given database. This filtering functionality can be valuable in replication environments where Directory Servers are separated by WANs where one of your major objectives is to keep replication costs to a minimum.

CAUTION Fractional replication is not backward compatible with versions of Directory Server prior to Directory Server 5.2. Therefore, ensure that all other instances of Directory Server are of the following versions: Sun ONE Directory Server 5.2, or Sun Java Enterprise System Directory Server 5.2 2003Q4 or later.

Business Scenarios and Their Associated Replication Solutions

Now that you are familiar with basic replication concepts and configurations we will examine three different business scenarios with their associated replication solutions, to provide you with an initial understanding of how you might choose to deploy the replication possibilities Directory Server provides. This section will present the following scenarios:

- Small Site with Local Availability Imperatives
- Large Site with Heavy Network Traffic

- International Site with Data Privacy, High Availability and Performance Imperatives

Small Site with Local Availability Imperatives

Suppose your entire enterprise is contained within a single building. This building has a fast (100 MB per second) and lightly used network. The network is stable and you are reasonably confident of the reliability of your server hardware and OS platforms. You are also sure that a single server's performance will easily handle your site's load. However, as an online retail business, albeit a small business, high availability is crucial to your business success.

In this case, replicate at least once to ensure availability when your primary server is shut down for maintenance or hardware upgrades. Also, set up a DNS round robin to improve LDAP connection performance in the event that one of your Directory Servers becomes unavailable. Alternatively, use an LDAP proxy such as Sun Java System Directory Proxy Server. For more information on Directory Proxy Server, see `http://www.sun.com/software/products/directory_proxy/home_dir_proxy.html`.

Large Site with Heavy Network Traffic

Suppose your entire enterprise is contained within two buildings. The network is stable and you are reasonably confident of the reliability of your server hardware and OS platforms, and that a single server's performance will easily handle the load placed on a server within each building. However, the network has to support heavy loads, and is often saturated during normal working hours.

Also assume that you have slow (ISDN) connections between the buildings, and that this connection is very busy during normal business hours. Fortunately, however, there is no real need for close data consistency.

A typical replication strategy for this scenario would be:

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

  This server should be placed in the building that contains the largest number of people responsible for the master copy of the data. Call this Building A. Choosing Building A to house the master copy of the data should help you to ensure that an increased number update operations are likely to be made from the same building, thus taking advantage of a faster and more reliable network connection.

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

  Use multi-master replication to ensure write-failover.

- Create two replicas in the second building (Building B).
- As 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, to alleviate network traffic loads.
- Configure cascading replication in each building with an increased number of consumers dedicated to lookups on the local data to provide further load balancing.

International Site with Data Privacy, High Availability and Performance Imperatives

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

A typical replication strategy for this scenario would be:

- Hold master copies of directory data on servers in both data centers.
- Deploy a fully meshed, four-way, multi-master replication topology between France and the USA to provide high-availability and write-failover across the deployment.

  Write-failover in both sites is essential to your 24x7 availability needs, and despite the performance strain it may place on your system, is not negotiable. The replication performance enhancements afforded by Directory Server, render it possible to configure a fully meshed, four-way, multi-master replication topology over the WAN. Although this topology threatens to place a strain on the system, the fact that you have a master copy of the data in each location, will prevent local users from having to perform update and lookup operations over the dial-up connection, thus optimizing performance.

  To safeguard your high-availability needs ensure that the replication agreements you configure between the two French and USA master pairs are configured over separate network links to guard against the eventuality of one of the network links becoming unavailable or unreliable.
- Deploy as many consumers as you require in each data center to reduce the load on your masters in terms of directory lookups.
- Set up fractional replication agreements between masters and consumers in both geographical locations, to filter out the data you do not wish your partners to access.

- Schedule replication so that it occurs only during off peak hours to optimize bandwidth.

Backup and Restoration Possibilities

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 have to re-initialize a failed master from another master. This section briefly presents the tools available to you for backing up and restoring your data. The command line utilities presented in this chapter are subcommands of the `directoryserver` command. For more information, see the *Directory Server Man Page Reference*. Based on the advantages and limitations of each method, you will select what best suits your overall requirements. This section is divided into two parts:

- Backup Methods
- Restoration Methods

Backup Methods

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. It does *not* back up the `dse.ldif` configuration file, which you will need to do manually to restore a previous configuration.

Performing a binary backup allows you to back up all suffixes at once, and is significantly faster than a backup to LDIF. However, binary backup can only be performed on a server with an *identical* configuration. For the detailed list of what an *identical* configuration implies see the Planning a Backup Strategy section in the *Directory Server Deployment Planning Guide*. At a minimum, 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, perform a binary backup on each server.

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. Note that replication information is only backed up if you use the `-r` option when running `db2ldif` and that the `dse.ldif` configuration file is not backed up in a backup to LDIF. Back this file up manually to enable you to restore a previous configuration.

Backup to LDIF has the advantage of being able to be performed from any server regardless of its configuration. However, in situations where rapid backup and restoration are required, backup to LDIF may take too long to be viable.

Take time to formulate a backup strategy which is well adapted to your availability needs. For further restrictions related to backups and replication see the Planning a Backup Strategy section in the *Directory Server Deployment Planning Guide*.

Restoration Methods

Directory Server also 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. Again it will be up to you to decide which best suits your overall requirements.

Binary Restore (bak2db)

Binary restore copies data at the database level. Restoring data using binary restore therefore has the advantage that all suffixes can be restored at once and that it is significantly faster than restoring from an LDIF file. However, restoring data using binary restore can only be performed on a server with an *identical* configuration, and in the event of you being 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.

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 advantage that it can be performed on any server, regardless of its configuration, and that it allows you to renew all the indexes (particularly useful where existing indexes are corrupt). Because a single LDIF file can be used to deploy an entire directory service, regardless of its replication topology, restoration from LDIF is particularly useful for the dynamic expansion and contraction of a directory service according to anticipated business needs. However, in situations where rapid restoration is required, restoration from an LDIF file may take too long to be viable.

High Availability Support

The Sun Cluster agent is a high availability agent that provides LDAP service failover. It is bundled with Directory Server and on certain hardware and deployment configurations, can be used to improve the availability of Sun Java System services provided by multiple servers.

With Cluster, one backup node is offline, but constantly checks the viability of the primary node. If it does not respond, the backup node takes over the IP identity of the original node, responding to operations requests. This feature is useful for directories that support large populations with strict availability requirements, such as large banks or telecommunication companies. For more information on Sun Cluster see *Sun Cluster 3.1 Product Documentation*.

Chapter 5

Directory Server Scalability

This chapter presents what Directory Server has to offer in terms of product scalability. We examine what features make Directory Server scalable and this chapter is divided accordingly into the following sections:

- Multiple Database, Multiple Server, and Data Distribution Possibilities
- Scalable Data Management
- Directory Proxy Server and Scalability

Multiple Database, Multiple Server, and Data Distribution Possibilities

One obvious prerequisite for scalability is the ability to distribute your data across multiple databases and server instances. This section addresses the following topics:

- Distributing Data Across Multiple Databases and Servers
- How to Manage Your Distributed Data

Distributing Data Across Multiple Databases and Servers

Distributing data enables you to scale your directory across multiple databases and server instances. The server instances may or may not (depending on performance requirements) be stored on several machines. A distributed directory can therefore hold a much larger number of entries than would be possible with a single server. What is more, the distribution details can be hidden from the user of client application, which means that as far as directory clients are concerned, a single directory answers their directory queries.

Directory Server's multiple database architecture renders these scalable distributed naming contexts possible, thus providing the ability to support millions of users on a single system, improve backup and restore, load balancing, and simplify administration. Add to this the fact that Directory Server can run as a 64-bit application on Solaris SPARC, which allows you to attain improved performance for high-volume deployments, and you can appreciate the considerable scalability potential Directory Server has to offer.

Depending on what your directory tree is like you may want to distribute it not only across separate databases, but have multiple databases to hold certain large branches of your directory tree. The database functionality provided by Directory Server provides flexible scalability in that you can:

- add databases dynamically without having to take the entire directory off-line
- view databases independently for ease of database management
- configure databases independently (referrals or not, attribute encryption or not, indexes) for finer granularity, irrespective of how large your deployment might be

If these databases are then distributed across multiple servers, which in turn generally tends to equate to several physical machines, we can see that data distribution reduces the work each server needs to do, thus allowing the directory to scale to a much larger number of entries than would otherwise be possible.

How to Manage Your Distributed Data

Once your data is distributed, you must then define the relationships between the distributed data. You do this using pointers to directory information held in different databases. Directory Server provides the referral and chaining mechanisms to help you link distributed data into a single directory tree:

- Referral

  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 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.

Chaining reduces client complexity, affords dynamic management possibilities, in that it allows you to remove a part of the directory from the system while the entire system remains available to client applications, and provides access control functionality, when the chained suffix impersonates the client application, providing the appropriate authorization identity to the remote server. However, this increased functionality does not come without increased server complexity costs.

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 method, or combination of methods, you choose will depend on the specific needs of your directory. For more detail on referrals and chaining, and how to decide between the two see the Referrals and Chaining section in the *Directory Server Deployment Planning Guide*.

Scalable Data Management

For data management functionality to satisfy enterprise directory service demands, scalability is essential. To prevent data management from becoming an onerous task Directory Server provides the following scalability features:

- Scalable Grouping of Entries
- Scalable Attribute Management
- Effective Rights Management
- Scalable Schema Management

Scalable Grouping of Entries

In addition to the hierarchical grouping mechanism provided by the directory information tree, which is of course not optimal for associations between dispersed entries, frequently changing organizations, or data repeated in many entries, Directory Server provides two features to ease identity and relationship management in large directory deployments: groups and roles. Both mechanisms allow for more flexible and thus scalable data management. We examine them both in the following sections:

- Groups
- Roles

• Tailoring Grouping Mechanisms To Scalability Requirements

Groups

A group is an entry that identifies the other entries that are its members, whose scope can encompass the entire directory. There are two types of groups, static and dynamic. Static groups explicitly name their member entries and are suitable for groups with relatively few members, such as your directory administrators. Dynamic groups, on the other hand, specify a filter and all entries that match the filter are then considered to be members of that group; dynamic because membership is defined each time the filter is evaluated. It is possible by using the DN of another group as member attribute of a dynamic group, to place groups inside other groups.

Roles

Roles are the second entry grouping mechanism that enable you to determine role membership as soon as an entry is retrieved from the directory. Each role has members, or entries that possess the role. Every entry that belongs to a role is given the `nsRole` virtual attribute whose values are the DNs of all roles for which the entry is a member (termed virtual since it is generated on-the-fly by the server and not stored in the directory). There are three different types of roles:

- Managed Roles - Explicitly assign a role to member entries
- Filtered Roles - Entries are members if they match a specified LDAP filter
- Nested Roles - Roles which contain other roles

which allows you to specify roles either explicitly or dynamically depending on which type of role you use.

Tailoring Grouping Mechanisms To Scalability Requirements

Because the groups and roles mechanisms provide a degree of overlapping functionality, it is important to understand their advantages and disadvantages in order to optimize the degree of scalability they can both provide. Generally speaking, the more recent roles mechanism is designed to provide frequently required functionality more efficiently. However, because the choice of a grouping mechanism influences server complexity and determines how clients process membership information, you must plan your grouping mechanism carefully. You must have a detailed understanding of the typical set membership queries and set management operations you will need to perform, to be able to decide which mechanism is more suitable.

Favor the groups mechanism if your priorities include:

- Obtaining a membership list, when the size of the group is less than 20,000 members (above this size of group, Directory Server will perform better using roles)
- Assigning and removing members (since no special access rights are required to add the user to the group contrary to roles)
- Minimizing computation overheads for performance reasons
- Adding or removing sets into or from existing sets (as the groups mechanism does not have any of the nesting restrictions that roles do)
- Maintaining flexibility of scope for grouping entries (roles can only extend their scope via a nested role on the *same* server instance)
- Using the chaining functionality to distribute your data (as roles are subject to chaining restrictions)
- Supporting LDAP client applications which require group entries to be of the `groupOfNames` or `groupOfUniqueNames` object classes

Likewise, favor the roles mechanism if your priorities include:

- Rapidly enumerating the members of a set and calculating set membership for an entry

  (Roles push membership information out to the user entry where it can be cached to make subsequent membership tests more efficient, the server performs all computations, and the client only needs to read the values of the `nsRole` attribute. In addition, all types of roles appear in this attribute, allowing the client to process all roles uniformly. Roles can perform both operations more efficiently and with simpler clients than is possible with groups).
- Integrating your grouping mechanism with existing Directory Server functionality such as Password Policy, Account Inactivation and Access Control thus taking advantage of the role membership computations that the server will do automatically.

For a list of other implementation considerations surrounding roles and groups refer to the “Grouping Directory Entries” and “Managing Attribute” sections of the *Directory Server Deployment Planning Guide*.

Scalable Attribute Management

Directory Server also provides an attribute management mechanism called Class of Service (CoS) for sharing attributes between entries in a way that is invisible to applications. With CoS, some attribute values need not be stored with the entry itself. Instead they are generated by the CoS logic as the entry is sent to the client application. For the client application, these attributes appear just like all other attributes. CoS allows related entries to share data for coherence and space considerations. We examine the following in this section:

- About CoS
- CoS Implementation Considerations

About CoS

Imagine a directory containing thousands of entries that all have the same value for the `facsimileTelephoneNumber` attribute. Traditionally, to change the fax number, you would update each entry individually, a time consuming job for administrators. Using CoS, the fax number is stored in a single place, and Directory Server automatically generates the `facsimileTelephoneNumber` attribute on every concerned entry as it is returned. It goes without saying that CoS is crucial to the scalability of your directory deployment, as it means that directory administrators only have a single fax value to manage as opposed to the thousands that would otherwise be the case. There are three different types of CoS which provide three slightly different types of attribute management functionality. For a detailed insight into the different types of CoS see theManaging Attributes with Class of Service section in the *Directory Server Deployment Planning Guide*.

Additional scalability is brought by the fact that generated CoS attributes can be multi-valued, roles and CoS can be used together to provide role-based attributes, and CoS priorities can be set (should CoS schemes be created that actually compete with each other to provide an attribute value). Scalability is enhanced given that CoS can bring:

- Improved manageability
- Disk space savings (fewer values are actually stored in the directory)
- Simplified client access to information and eased administration of repeated attributes

CoS Implementation Considerations

Bear in mind that as CoS virtual attributes are not indexed, referencing them in an LDAP search filter may have an impact on performance, and that as CoS schemes are resource intensive, they should only be used when strictly necessary. As the server caches CoS information, modifications to CoS definitions do not take

immediate effect, and as a result, frequent CoS modifications do present a risk, during cache reconstruction, of outdated data being accessed. The cache management performance improvements brought by Sun Java System Directory Server 5.2 2005Q1 do not completely eliminate this risk. For other implementation considerations related to access control and distribution, see “CoS Limitations” in the *Directory Server Deployment Planning Guide* for more detail regarding CoS related limitations.

Effective Rights Management

The access control model provided by Directory Server is powerful in that access can be granted to users via many different mechanisms. However, this flexibility can make determining what your security policy comprises fairly complex. It is important for this powerful access control model to remain manageable, that is, for it to be simple to administer users and debug the access control policy in place, if it is to scale to the complex deployment requirements at stake, which renders an effective rights retrieval functionality essential. This ability to request the rights of a given user to directory entries and attributes is provided by the Effective Rights LDAP control supported by both the console and LDAP search.

When using the control you specify the DN of the user for which you want to know the effective rights, as well as any additional attributes. The information returned includes:

- Rights information including entry level rights, attribute level rights and logging
- Logging information, which helps you debug access control problems

It is important to note that this rights information corresponds to the ACIs effective at the time of your request, the authentication method used, and the host machine name and address from which your request was made. Also note that viewing effective rights is itself a directory operation that should be protected by placing ACIs on the effective rights attributes. For further information regarding the Effective Rights functionality refer to the “Requesting Effective Rights Information” section of the *Directory Server Deployment Planning Guide*.

Scalable Schema Management

Directory Server comes with a standard schema that includes hundreds of object classes and attributes. In terms of scalable schema management Directory Server provides you with the following functionality:

- Online Schema Extension

  Not only can you extend existing schema by creating new object classes and attributes, either using the Console or using basic LDAP modify commands (from the command line or from an LDAP enabled application), but this can be done without having to take the entire directory off-line.

- Schema Checking

  This schema checking functionality allows you to ensure that the object classes and attributes you are using are defined in the directory schema, that the attributes required for an object class are contained in the entry, and that only attributes allowed by the object class are contained in the entry (thus avoiding any resulting inconsistencies).

Directory Proxy Server and Scalability

Directory Proxy Server brings additional scalability to Directory Server which is often essential to the success of your directory service. A brief outline of what Directory Proxy Server's can offer in terms of scalability follows.

Directory Proxy Server helps Directory Server to scale in that it is designed to support high availability directory deployments by providing both automatic load balancing and automatic fail over and fail back among a set of replicated LDAP Directory Servers. When deployments scale in both size and complexity, this guarantee of automatic load balancing, failover and failback is welcome to say the least.

Functionally, Directory Proxy Server is an "LDAP access router" located between LDAP clients and LDAP Directory Servers. Requests from LDAP clients can be filtered and routed to LDAP Directory Servers based on rules defined in the Directory Proxy Server configuration. Results from Directory Server can be filtered and passed back to clients, again based on rules defined in the Directory Proxy Server configuration. This process is totally transparent to the LDAP clients, which connect to Directory Proxy Server just as they would to any LDAP Directory Server.

For extranet and intranet environments it is often necessary to ensure that mission-critical directory-enabled clients and applications have 24x7 access to directory data. Directory Proxy Server maintains connection state information for all Directory Servers that it knows about, and is able to dynamically perform

proportional load balancing of LDAP operations across a set of configured Directory Servers. Should one or more Directory Servers become unavailable, the load is proportionally redistributed among the remaining servers. When a Directory Server comes back on line, the load is proportionally reallocated.

It is important to note however, that to avoid Directory Proxy Server becoming the single point of failure for your directory deployment, you should use at least two Directory Proxy Servers with an IP appliance in front of them.

Chapter 6

Directory Server Security

This chapter examines the crucial aspect of security and what functionality Directory Server provides to address the principle threats to directory security, which are generally either related to unauthorized access, unauthorized tampering or denial of service attacks. This chapter is divided into the following sections:

- Directory Server Security Overview
- Authentication and Account Inactivation
- Controlling Access
- Password Policy
- Securing Connections and Data Storage
- Denial of Service Attacks

Directory Server Security Overview

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

- 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.

- Account inactivation

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

- Password Policy

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

- Encryption

A means of protecting sensitive data by scrambling it in a way that only authorized users have the means of decrypting.

- Access control

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

- 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, a basic audit procedure is to automate testing using a network-based security scanner that identifies vulnerabilities on your network. You can also conduct audits by examining the log files and the information recorded by the SNMP agents.

Authentication and Account Inactivation

While it may seem simple to protect your directory from unauthorized access, the problem can in fact be extremely complex. To have a secure directory it is necessary to protect against unauthorized access which can occur both from inside your company and outside your company if it is connected to an extranet or to the Internet.

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

For more detail on these authentication mechanisms see the Selecting Appropriate Authentication Method section in the *Directory Server Deployment Planning Guide.*

The account inactivation functionality in Directory Server, which allows you to temporarily inactivate a user account or a set of accounts, is essential to your directory security. Once inactivated, a user cannot bind to Directory Server, and the authentication operation fails. Account inactivation is implemented through an account lock attribute, which, when enabled, causes the server to automatically reject the bind operation.

Account Inactivation can be combined with the roles functionality to enable you to inactivate all members of a given role should that kind of account inactivation become necessary. The combined use of roles and account inactivation provides scalable security functionality, a must in the enterprise context.

Controlling Access

Controlling access enables you to specify that certain clients have access to particular information, while other clients do not. You implement access control using one or more access control lists (ACLs). ACLs consist of a series of access control instructions (ACIs) that either allow or deny permissions (such as read, write, search, proxy, add, delete, and compare) to specified entries and their attributes. Directory Server's access control mechanism provides fine-grained, flexible access control in that it allows you to set permissions for either:

- a network location such as an IP address or a DNS name
- the entire directory
- a particular subtree of the directory
- specific entries in the directory
- a specific set of entry attributes
- or any entry that matches a given LDAP search filter

Add to this fine-grained, flexible model the fact that you can set permissions for a specific user, for all users belonging to a set (be it a group or a role), or for all users of the directory and you understand how powerful Directory Server's access control model is. This section examines the following:

- Access Control - The Basics
- Macro ACIs
- Requesting Effective Rights Information
- ACIs - The Tips

Access Control - The Basics

ACIs are stored in the directory, as attributes of entries. The aci attribute is an attribute which is available for use on every entry in the directory and 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 can be returned in an `ldapsearch` operation if specifically requested. A sample aci attribute follows:

```
aci: (target)(version 3.0;acl "name";permission bindRules;)
```

where

- `target` which specifies the entry, attributes, or set of entries and attributes for which you want to control access
- `version 3.0` is a required string which identifies the ACI version
- `name` is a name which is a name for the ACI. This ACI name is required and should describe the effect of the ACI
- `permission` is a permission which specifically states what rights the ACI either allows or denies
- `bindRules` specifies the credentials and bind parameters that a user has to provide to be granted access.

When you install Directory Server, or when you add a new suffix, a number of default ACIs are defined. These ACIs can be modified to suit your security requirements. The default policy is not to grant users access rights, which means that you must begin by setting some ACIs to enable your users to access the directory. The access control mechanism precedence rule states that when two conflicting permissions exist the permission that denies access always takes precedence over the permission that grants access, which means that you need to be careful when explicitly denying access that you do not end up denying more access than you actually intended.

One of the more powerful features of the Directory Server access control model is the ability to use LDAP search filters to set access control. LDAP search filters enable you to set access to any directory entry that matches a set of defined criteria. For an example of how this might work see the Setting Permissions section of the *Directory Server Deployment Planning Guide*.

Macro ACIs

Directory Server provides the Macro ACI functionality for optimizing the number of ACIs in your directory, which is particularly useful given that ACIs can have a noticeable impact on memory consumption. Macro ACIs are placeholders that are used to represent a DN, or a portion of a DN, in an ACI. You can use a macro to represent a DN in the target portion of the ACI, in the bind rule portion, or both. When Directory Server receives an LDAP request, the macro ACIs are matched against the resource targeted by the LDAP operation. If there is a match, the macro is replaced by the value of the DN of the targeted resource. Directory Server then evaluates the ACI normally.

Requesting Effective Rights Information

This ability to request the rights of a given user to directory entries and attributes is provided by the Effective Rights LDAP control supported by both the console and LDAP search. This Effective Rights functionality helps you administer and debug Directory Server's powerful access control mechanism, which in the enterprise and E-business context is sometimes no mean task. See the section "Effective Rights Management" on page 61 for further information.

ACIs - The Tips

There are many simple tips which can help facilitate access control management and improve directory performance. Although many of them constitute basic common sense such as keeping the number of ACIs in your directory to a minimum and naming them meaningfully, it is important to bear these tips in mind when implementing access control. For a complete list of ACI tips refer to the "Tips on Using ACIs" section in the *Directory Server Deployment Planning Guide*.

Password Policy

- Password Policy Overview
- Configuring Password Policies
- Preventing Dictionary-Style Attacks
- Controls for Resetting Passwords

Password Policy Overview

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

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

- User-Defined Passwords - whether or not your password policy allows for user-defined passwords.
- Password Change at First Login or Reset - whether or not your password policy forces users to change their password at the first login or after a password has been reset by an administrator.
- Password Expiration - whether or not passwords can be used indefinitely or expire after a given time. By default, user passwords never expire.
- Expiration Warning - specifies when you send out your password expiration warning.
- Password Syntax Checking - whether or not your password policy enforces use of the syntax checking mechanism, to ensure password string compliance with guidelines.
- Password Length - specifies the minimum length for user passwords.
- Password Minimum Age - specifies whether your password policy prevents users from changing their passwords within a specified period, which can be used in conjunction with the password history feature to discourage users from reusing old passwords.
- Password History - specifies the maximum number of passwords that can be stored in the password history.
- Password Storage Scheme - specifies the type of encryption used to store passwords within the directory, be it clear text (no encryption), Secure Hash Algorithm (SHA), Salted Secure Hash Algorithm (SSHA), or UNIX CRYPT algorithm.

For a more detailed insight into each of these password policy features, refer to the “Password Policy Feature” section in the *Directory Server Deployment Planning Guide*.

Configuring Password Policies

In previous releases of Directory Server, the only password policy that existed was a global policy (see the Global Password Policy section of the *Directory Server Deployment Planning Guide* for the default values), which although it could be adapted, meant that the same password policy had to be applied across the board. This global password policy was backed up by the hard-coded password policy, whenever the global password policy was either absent or, following modifications, no longer valid.

Now Directory Server allows the user to configure multiple password policies and choose to assign them either to particular users or to whole sets of users using the CoS and roles functionality. This flexible password policy scheme allows you to create password policies that meet security requirements for specific users or roles.

For example, an ISP decides it needs three password policies. Clients have passwords that never need to be changed. Internally, employees have passwords that they must change every month. Administrators, however, have to change their password every week, and the password must be 8 characters in length. Using the multiple password policy feature, each of these policies can be assigned to users using roles and class of service.

This new password policy design provides you maximum flexibility for adapting the Directory Server to the needs of your business model. In this multiple password policy context, it is worth noting that the following order of precedence applies:

1. A password policy generated by a CoS definition will take precedence over a password policy assigned directly to the user entry.
2. A password policy assigned directly to a user entry will take precedence over the global password policy.
3. The global password policy, stored under `cn=Password Policy,cn=config`, will take precedence over the hard-coded password policy values provided with Directory Server.

CAUTION There are certain considerations to be taken into account regarding password policies in replicated environments. For more information see the Password Policies in a Replicated Environment section of the *Directory Server Deployment Planning Guide*.

Preventing Dictionary-Style Attacks

In a dictionary-style attack, an intruder attempts to crack a password by repeatedly guessing the password until gaining authorization. The server provides three tools to help counter such attacks:

- Password syntax checking verifies that a password does not match values of the `uid`, `cn`, `sn`, `givenName`, `ou`, or `mail` attributes of the user entry. The server will not allow a user to set a password if it matches one of these values. However, syntax checking does not thwart actual dictionary attacks, in which the intruder tries every word in `/usr/dict/words`, for example.
- A minimum password length ensures that the user cannot set a short password. Passwords with more characters are exponentially harder to guess or attempt all values.
- The account lockout mechanism prevents binding after a certain number of failed authentication attempts. The lockout may either be temporary or permanent, depending on how strict you wish to make your password policy. Note that 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.

Controls for Resetting Passwords

Passwords can be reset using a new modification request control that checks an ACI to confirm that the client binding to the directory is allowed to make the change. The directory manager can configure an administrator group with members who have the appropriate privileges to update passwords.

Securing Connections and Data Storage

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. If intruders gain access to your directory or intercept communication between Directory Server and a client application, they have the potential to tamper with either directory data or directory configuration 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. Directory Server allows you to secure your connections and data storage and this section examines these areas in more detail:

- SSL Encryption and Authentication

- Start Transport Layer Security
- SASL Encryption and Authentication
- Attribute Encryption

SSL Encryption and Authentication

It is essential to protect the integrity of the information in transit over the network between servers and client applications. In order to respond to this security necessity Directory Server supports LDAPS, the standard LDAP protocol that runs on top of Secure Sockets Layer (SSL). SSL provides encrypted communications and optional authentication between Directory Server and its clients.

SSL may be enabled for both the LDAP and DSML / HTTP protocols to provide security for any connection to the server. You can use an SSL connection to bind to the server in combination with certificate-based authentication, providing higher levels of security than password checks alone. You can also configure replication to use SSL for secure communications between servers, preventing anyone from snooping your data. After a secure connection has been established, the SSL protocol requires the server to send its certificate to the client. Using public-key cryptography, the client can determine the authenticity of the certificate and verify that it was issued by a certificate authority that the client trusts. By verifying the certificate, the client can prevent a man-in-the-middle impersonation of the server by a third party.

Directory Server is capable of simultaneous SSL and non-SSL communications on separate ports. However, for security reasons, you may also restrict all communications to the secure port.

For further detail on the encryption and hashing algorithms (or ciphers), Directory Server can use see the Securing Connections With SSL section in the *Directory Server Deployment Planning Guide* . SSL uses encryption for privacy and hashing of checksums for data integrity. For background information regarding the use of SSL and TLS, see the *Administration Server Administration Guide*.

Start Transport Layer Security

Directory Server supports the Start Transport Layer Security (Start TLS) extended operation to enable TLS on an LDAP connection that was originally not encrypted. StartTLS is supported on Windows and UNIX platforms and consitutes a more recent version of the SSL protocol.

StartTLS allows you to have a secure connection even if you do not have a dedicated encrypted port. You can open a secure connection over the regular LDAP port.

For background information regarding the use of SSL and TLS, see the *Administration Server Administration Guide.*

SASL Encryption and Authentication

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. Directory Server supports authentication and encryption using the Simple Authentication and Security Layer (SASL). The SASL mechanism is responsible for authenticating a user through SASL credentials.

On Solaris systems, Directory Server supports the Generic Security Services API (GSSAPI) over SASL. The GSSAPI allows you to use a third-party security system such as Kerberos version 5 to authenticate clients. The server uses this API to validate the identity of the user. Then, the SASL mechanism applies the GSSAPI mapping rules to obtain a DN, which is the bind DN for all operations during this connection.

Client authentication may also be performed using DIGEST-MD5, a SASL-based mechanism of authenticating clients by comparing a hashed value the client sends with a hash of the user's password. Because the mechanism must read user passwords, you must use a reversible encryption method for the passwords of all users who want to be authenticated through DIGEST-MD5 or store these passwords in the directory in clear text.

SASL provides advanced security, allowing Directory Server to adapt to the security standards already established within your enterprise.

Attribute Encryption

As we have already seen 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 to protect data either when it is being stored in database files, backup files, and LDIF files, or being exported. 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 Directory Server attribute encryption feature provides this kind of data protection, crucial to your directory's security.

Attribute encryption is transparent to the user, who can perform LDAP operations without being aware of encryption. However, should you need to, it is possible to export data that remains encrypted, allowing for example, two servers to share secure files for updates.

The Directory Server attribute encryption feature is configured at database level, which means that once you decide to encrypt an attribute, that attribute will be encrypted for every entry in the database. However, it is worth noting that because attribute encryption occurs at an attribute level, the only way to encrypt an entire entry is to encrypt all of its attributes. The set of encryption algorithms provided by Directory Server can be used across different platforms to encrypt and decrypt attributes on disk, thus increasing the security of your data.

While attribute encryption offers increased data security, it does incur certain performance costs. Bearing this in mind, think carefully about which attributes require encryption and encrypt only those attributes you consider to be particularly sensitive. Note also that 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), configure attribute encryption *before* data is imported or added to the database for the first time, so that encrypted attributes are indexed as such from the outset.

For further detail on how attribute encryption is implemented and certain usage considerations, refer to the “Encrypting Attributes” section in the *Directory Server Deployment Planning Guide*.

Denial of Service Attacks

Being able to guard against denial of service attacks where the attacker’s goal is to prevent the directory from providing service to its clients simply by using the system’s available resources, is an essential area of security functionality. Directory Server allows you to set limits on the resources allocated to a particular bind DN, thus providing a protection against such attacks. For details on how to configure these protective limits see the Setting Individual Resource Limits section of the *Directory Server Administration Guide*.

Chapter 7

Directory Server Manageability

In the context of such a powerful product, manageability is crucial to the user and this chapter outlines what Directory Server provides in terms of product manageability. The chapter is divided into the following sections:

- Server Management Console
- Extensive Monitoring Possibilities

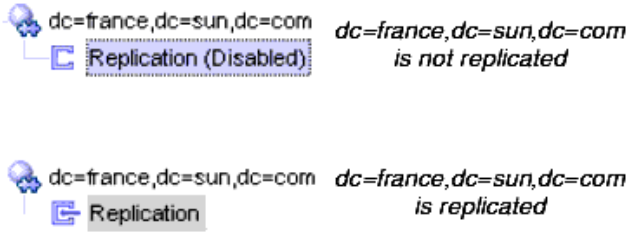
Server Management Console

Directory Server Console provides what users need in the context of such a powerful product, namely a user-friendly management console for performing administrative directory tasks, such as finding and adding entries, updating schema, creating and managing groups and roles, creating CoS, creating and managing access control instructions, inactivating user accounts or domains of accounts, creating replication agreements, and enabling replication. The *Directory Server Administration Guide* contains procedural information on how to use Directory Server Console to perform all administrative tasks you will need to perform throughout your directory deployment.

Directory Server Console also provides support for the assistive software and technologies that make software accessible to users with disabilities.

The user interface provided by Directory Server Console for managing replication serves as a good example of how user-friendly Directory Server Console can render complex configuration and management tasks. To begin with it provides the user with at a glance information about the status of replication, as each suffix node in the Console has a replication node attached to it, and in turn an icon indicating whether or not the suffix is replicated, as shown in Figure 7-1.

Figure 7-1 Replication Nodes in Directory Server Console With Their Replication Enabled and Replication Disabled Icons



When you click a replication node under a suffix in Directory Server Console, its contents provide you with different options depending on the state of the suffix as illustrated in Figure 7-2.

Figure 7-2 Replication Panel on the Directory Server Console Configuration Tab

In addition to this intuitive user interface Directory Server also provides a wizard for automatically creating replication agreements. When configuring and managing the replication topologies for large, enterprise deployments, this kind of manageability is very welcome.

Extensive Monitoring Possibilities

Another key area of manageability is the ability to monitor, because for your Directory Server to be manageable, you must be able to monitor its activity. Directory Server provides extensive monitoring possibilities and this section examines them in more detail:

- Command-Line Tools
- Directory Server Logs
- Directory Server Console
- Replication Discovery and Monitoring Tools
- Simple Network Management Protocol (SNMP)

Command-Line Tools

Command-line monitoring tools include operating system-specific tools to monitor performance such as disk usage, LDAP tools such as ldapsearch to collect server statistics stored in the directory, third party tools, or custom shell or Perl scripts.

Directory Server Logs

The access, audit, and error logs provided with Directory Server are a rich source of monitoring information. These logs can be monitored manually or parsed using custom scripts to extract monitoring information relevant to your deployment. For information on the scripts that can be used to access logging information, refer to the *Directory Server Resource Kit Tools Reference*. For information on viewing and configuring log files refer to “Monitoring Directory Server Using Log Files” in the *Directory Server Administration Guide*.

Directory Server Console

Directory Server Console enables you to monitor directory operations in real time, via a graphical user interface. The Console provides general server information, including a resource summary, current resource usage, connection status, and global database cache information. It also provides general database information such as the database type, status and entry cache statistics, cache information, and information relative to each index file within the database. In addition, the Console provides information relative to the connections and operations performed on each chained suffix.

Replication Discovery and Monitoring Tools

Directory Server provides replication management tools that allow you to monitor replication between servers. These tools simplify administration, particularly for complex Directory Server architectures and can help you avoid errors. In addition to these tools which are described below, new attributes have been added to the replication agreement entry to help you identify the changes that have been sent to a consumer.

The replication discovery and monitoring tools include:

insync

This tool checks if a master replica is synchronized with one or more consumer replicas, thus enabling you to manage potential conflicts between suppliers or even whole servers.

entrycmp

This tool compares a replicated entry to a copy of the entry on the consumer or master, thus enabling you to assess replication status.

repldisc

This tool depicts your complete replication topology starting from one server and building a graph all the known servers in the 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 the SNMP Monitoring section in the *Directory Server Deployment Planning Guide*. For information on how to set up and configure SNMP refer to "Monitoring Directory Server Using SNMP" in the *Directory Server Administration Guide*.

Glossary

Refer to the *Java Enterprise System Glossary* (http://docs.sun.com/doc/816-6873) for a complete list of terms that are used in this documentation set.

Index

A

access control 67
  administering and debugging 61
  instruction 67
  LDAP search filters 68
  list 67
  target 68
access control permission 68
account inactivation 67
ACI 67
  macro 69
ACL 67
anonymous access 66
approximate index 37
attributes 31, 35
  encryption of 74
  multi-valued 35
  operational 31
  scalable management of 60
  standard 35
authentication 65
  anonymous access 66
  certificate based client authentication 67
  proxy authorization 66
  SASL-Based Client Authentication 67
  simple password 66
  simple password over a secure connection 67

B

backing up data 51
bak2db 33, 52
base DN, ldapsearch and 32
binary restore 52

C

cascading replication 45
certificate-based client authentication 67
chaining 56
Class of Service 60
cn=config 29
conflict resolution 48
consumer 43
  initialization 44
consumer replica 42
CoS 60
Crypto Accelerator 1000 Board 40

D

data

backing up and restoring 33, 51
distributing 55
exporting 33
importing 32
indexing 33
integrity of 72
restoring 52
database
adding dynamically 56
configuring independently 56
viewing independently 56
datastore
optimized for reads 21
db2bak 33
db2bak-task 33
db2ldif 33, 52
db2ldif-task 33
dedicated consumer 43
denial of service attacks 75
dictionary style attack to security 72
directory information tree 29
Directory Proxy Server 62
Directory Server Console 77
directory service 19
distributed naming context 30
distributing data 55
DIT 29
DNS 20
documentation 9
domain component 30
dse.ldif configuration file 51
DSML 22
front end 26
DSRK 28

E

e-business directory 24
effective rights
LDAP control 61
management of 61
requesting 69
encrypting attributes 74
enterprise directory 23
enterprise-wide directory service 21
entries 31
entrycmp replication monitoring tool 80
equality index 37
extensible schema 21
online 62

F

failover 42
fault tolerance 42
filtered role 58
four-way MMR 47
fractional replication 48
front ends 26
fully connected replication topology 47

G

Generic Security Services API 74
grouping mechanisms 58
groups 58
dynamic 58
static 58
GSSAPI 74

H

hierarchical naming model 20
high availability 41, 48
hub 43, 45
hub replica 43

I

incremental update 44
indexes 33, 37
  approximate 37
  browsing 38
  equality 37
  international 38
  presence 37
  substring 37
  virtual list view 38
indexing data 33
insync replication monitoring tool 80
international index 38

J

JNDI 29

L

LDAP 22
  front end 26
  version 3 26
LDAP access router 62
LDAP SDK
  for C 28
  for Java 28
ldapsubentry 32
LDIF 31
  LDAP Data Interchange Format 11
ldif2db 33, 52
ldif2db-task 33
ldif2ldap 33
load balancing 42, 46
local data management 42
localizing data 42
logging 79

M

Macro ACIs 69
managed role 58
master replica 42
MMR 46
MMR over WAN 48
monitoring 79
  command-line tools 79
  console 79
  logs 79
  replication discovery and monitoring tools 79, 80
    entrycmp 80
    insync 80
    repldisc 80
multi-master replication 46
multiple database design 30, 56

N

nested role 58
NOS 23

O

o=NetscapeRoot 29
o=UserRoot 29
object class 36
online replica promotion and demotion 44
operational attributes 31
organizational unit 30

P

password
  change 70
  expiration 70
  expiration warning 70
  history 70

- length 70
- minimum age 70
- resetting 72
- storage scheme 70
- syntax checking 70
- user defined 70

password policy
- multiple policies 71
- overview 69

plug-ins 25
- directory server extensibility 27

presence index 37
proxy authorization 66

R

read-only replica 42
read-write replica 42
repldisc replication monitoring tool 80
replica
- consumer 42
- hub replica 43
- master 42
- read-only 42
- read-write 42

replica demotion 44
replica promotion 44
replication
- cascading 45
- concepts 41
- data consistency 44
- fractional 48
- monitoring 80
- multi-master 46
- single master 45

replication agreement 43
- disable 43, 47
- enable 43, 47

replication examples
- international sites 50
- large sites 49
- small sites 49

restore
- binary 52

restoring data 52
roles 58
- filtered 58
- managed 58
- nested 58

root suffix 29

S

SASL 74
SASL-based client authentication 67
schema 34
- checking 62
- format 34

search capabilities 21
secure connections 72
secure data storage 72
Secure Sockets Layer 73
security overview 65
shared network access 21
Simple Authentication and Security Layer 74
simple password authentication 66
simple password over a secure connection 67
single master replication 45
SNMP 25, 80
SSL 26, 73
- over TLS 26

Start TLS 73
Start Transport Layer Security 73
substring index 37
subtree 29
Sun Cluster Agent Support 31, 53
Sun Crypto Accelerator 1000 Board 40
supplier 43

T

total update 44

U

user id 30

V

virtual list view index 38
vlv index 38

X

X-ORIGIN field 34